cin.ufpe.br

Centro de Informática

U · F · P · E

Universidade Federal de Pernambuco

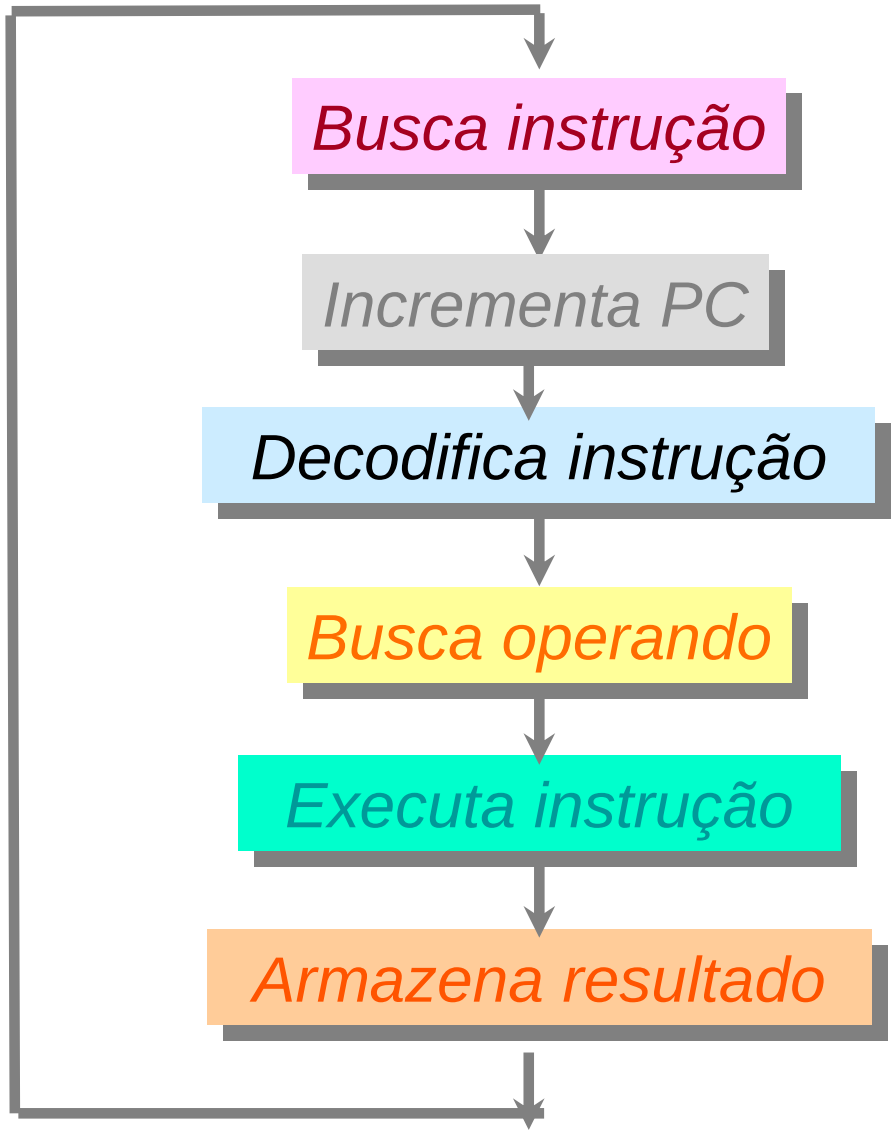
CPU: Estrutura e Funcionalidade

Implementação Multi-ciclo

Roteiro da Aula

- Projeto de uma CPU simples
- Unidade de Processamento – Via de Dados
  - Compartilhamento de unidades funcionais e memória
  - Registradores adicionais
  - Multiplexes
- Implementação Multi-ciclo
  - Unidade de controle
    - Leitura da Instrução
    - Operações Aritméticas
    - Leitura + Operação entre registradores
    - Acesso à Memória
    - Desvio Condicional
- Exceções
  - Instrução indefinida
  - overflow

Ciclo de Instrução

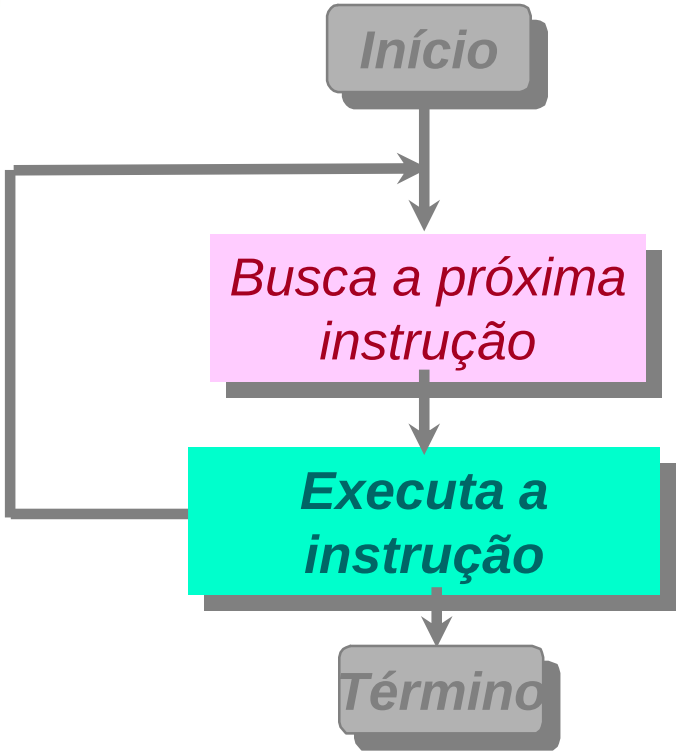


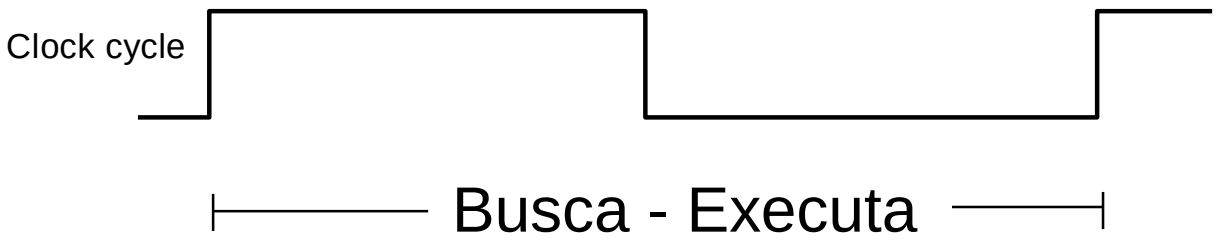
Projeto: uma CPU simples...

Instrução	Descrição
LD rt, desl(rs)	Carrega palavra de mem em rs
SD rt, desl(rs)	Armaz. Reg. na memória
ADD rd, rs, rt	rd <- rs + rt
SUB rd, rs, rt	rd <- rs – rt
AND rd, rs, rt	rd <- rs and rt
BEQ rs, rt, end	Desvio se rs = rt

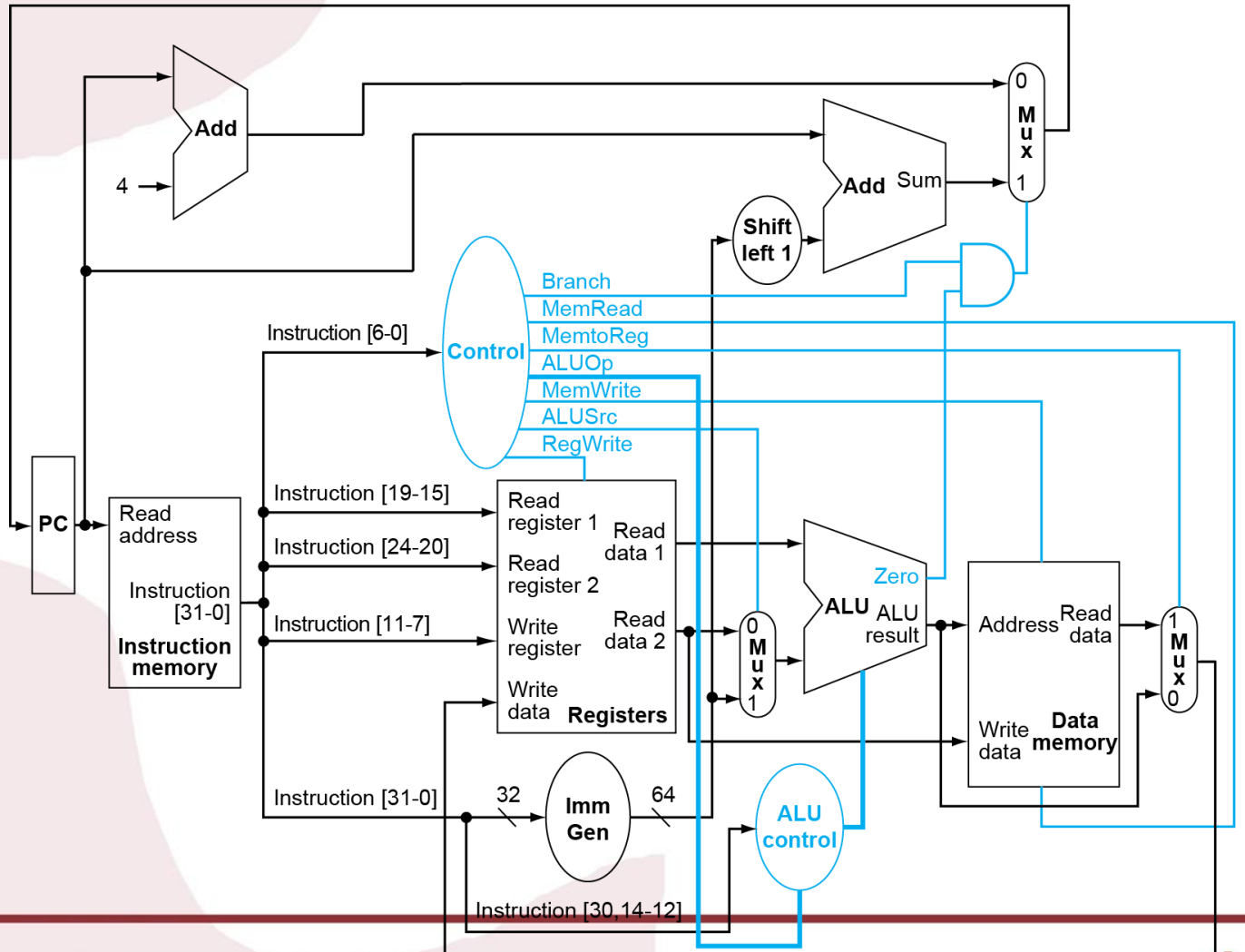
Name (Field Size)	Field 7 bits	5 bits	5 bits	3 bits	5 bits	7 bits	Comments
R-type	funct7	rs2	rs1	funct3	rd	opcode	Arithmetic instruction format
I-type	immediate[11:0]		rs1	funct3	rd	opcode	Loads & immediate arithmetic
S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode	Stores
SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode	Conditional branch format
UJ-type	immediate[20,10:1,11,19:12]				rd	opcode	Unconditional jump format
U-type	immediate[31:12]				rd	opcode	Upper immediate format

Mono-ciclo


Início
Busca a próxima instrução
Executa a instrução
Término


Clock cycle
Busca - Executa

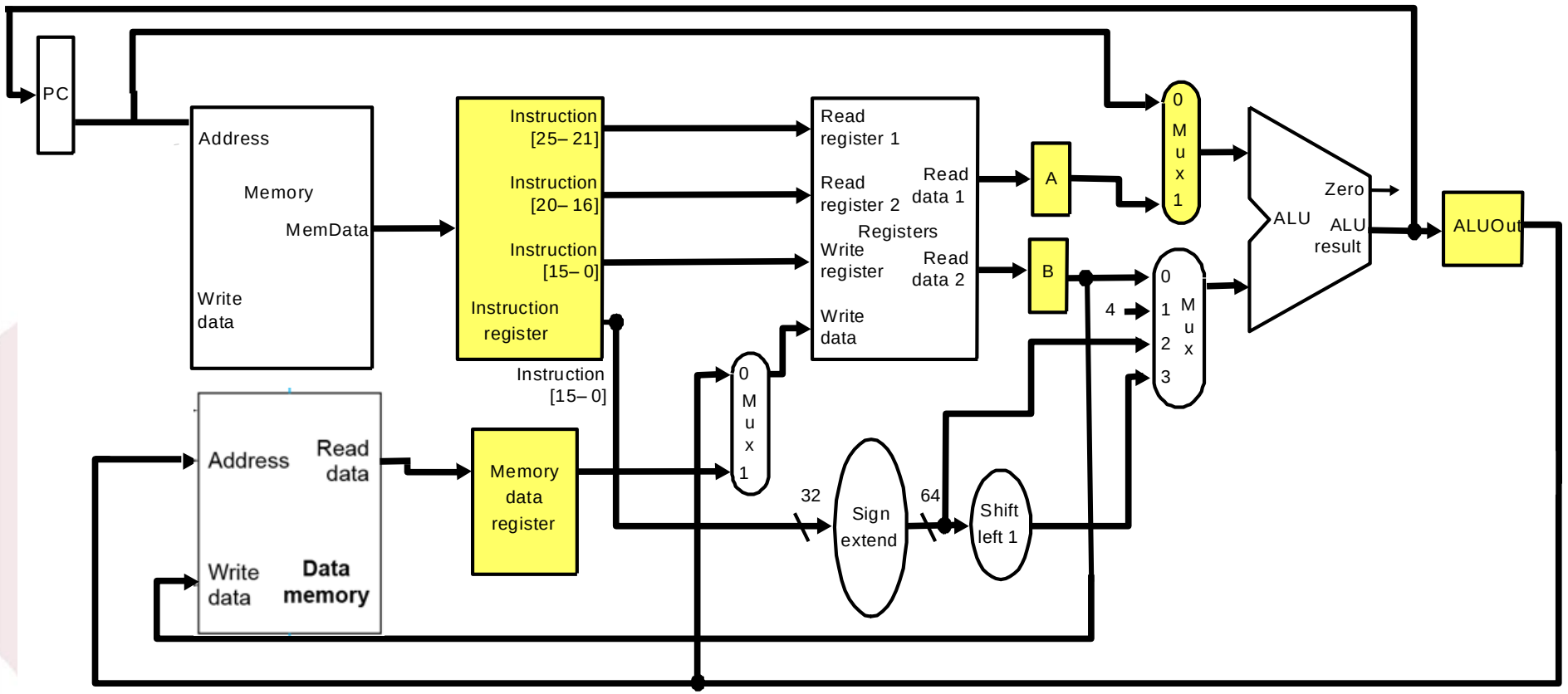
Mono-ciclo


Add
4
Add
Sum
Shift left 1
0
M
u
x
1
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [6-0]
PC
Read address
Instruction [31-0]
Instruction memory
Instruction [19-15]
Instruction [24-20]
Instruction [11-7]
Instruction [31-0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
32
Imm Gen
64
ALU control
Instruction [30,14-12]
Zero
ALU
ALU result
Address
Read data
Write data
Data memory
1
M
u
x
0

Implementação Multi-ciclo

Centro de Informática U·F·P·E

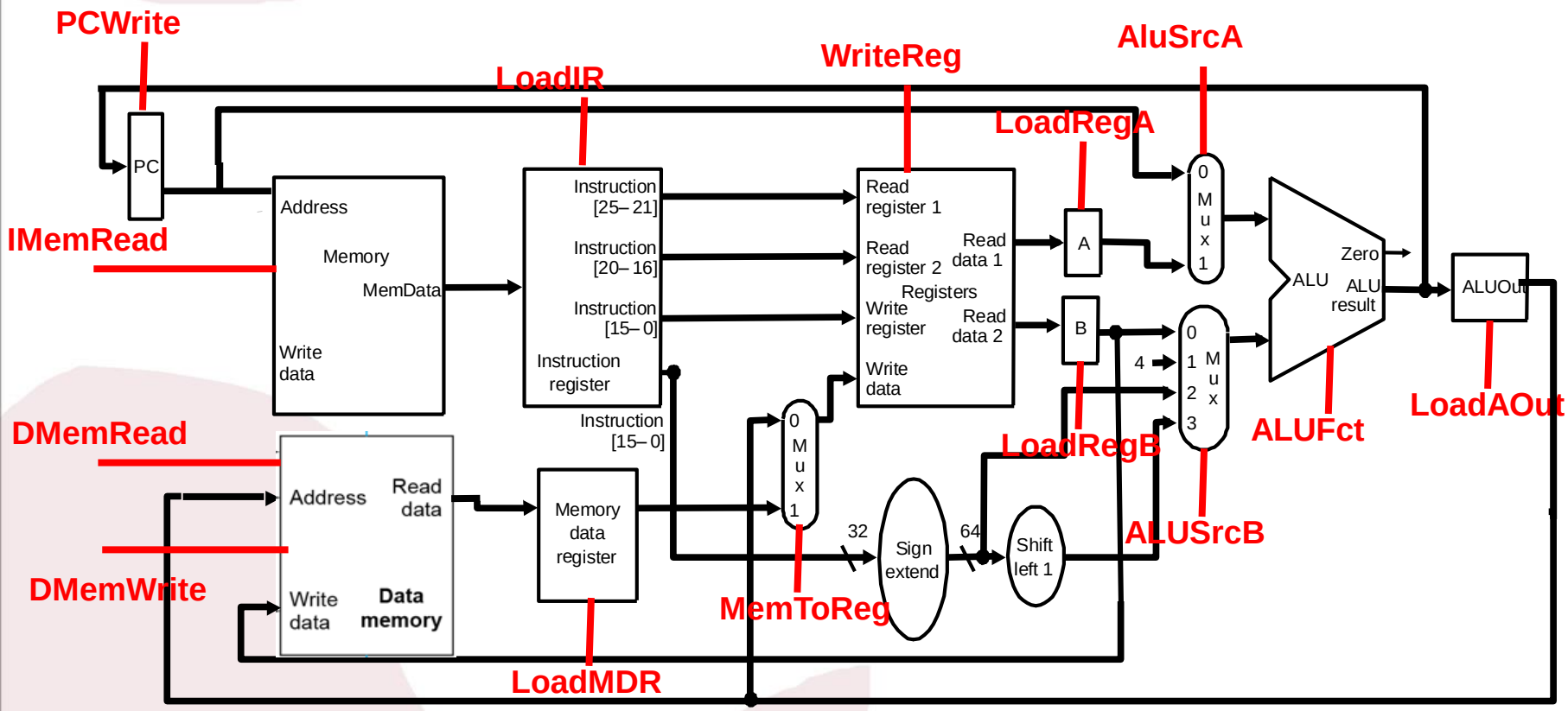
Unidade de Processamento – Via de Dados



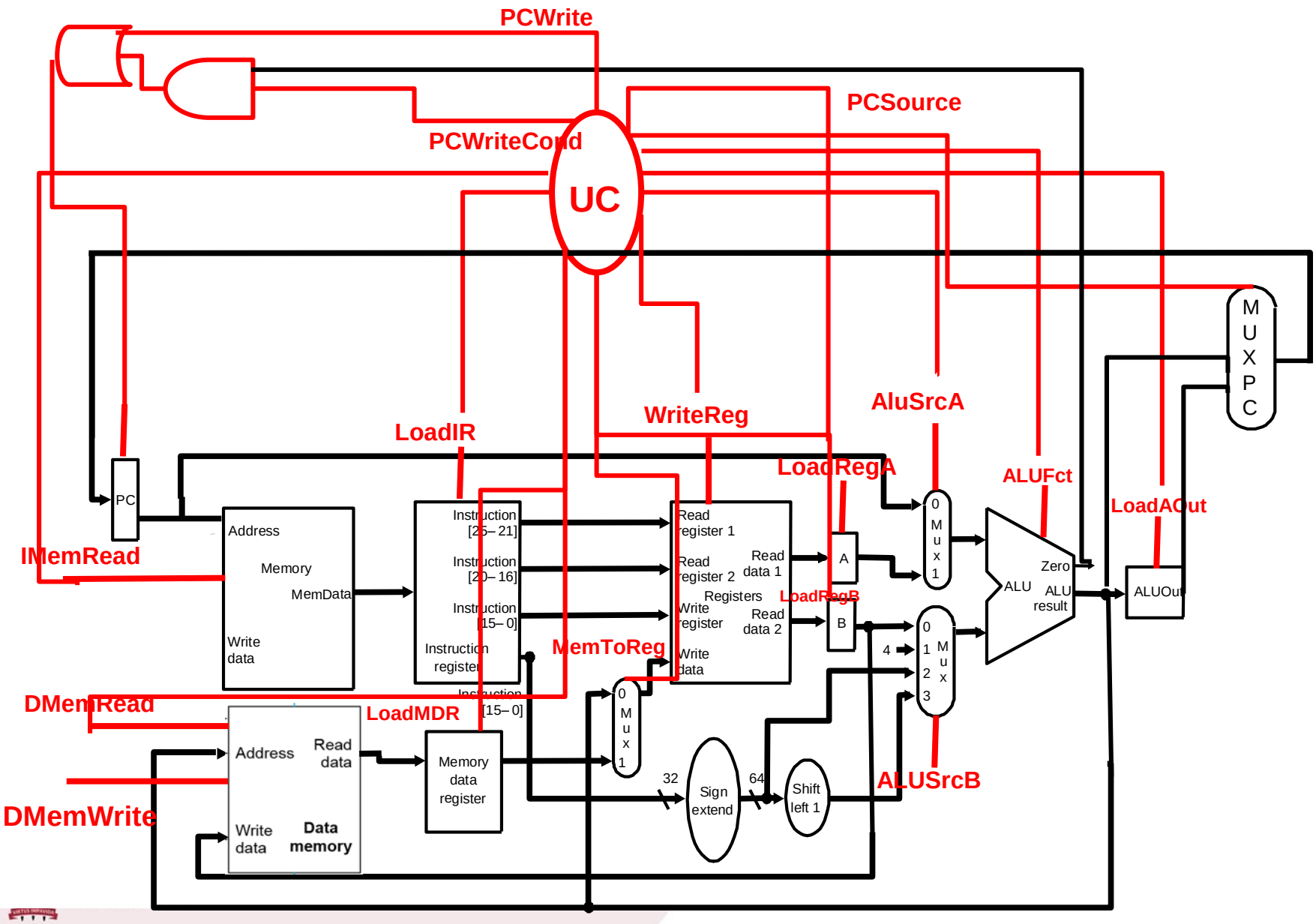
Implementação Multi-ciclo

Centro de Informática U·F·P·E

Unidade de Processamento – Via de Dados



Implementação Multi-ciclo



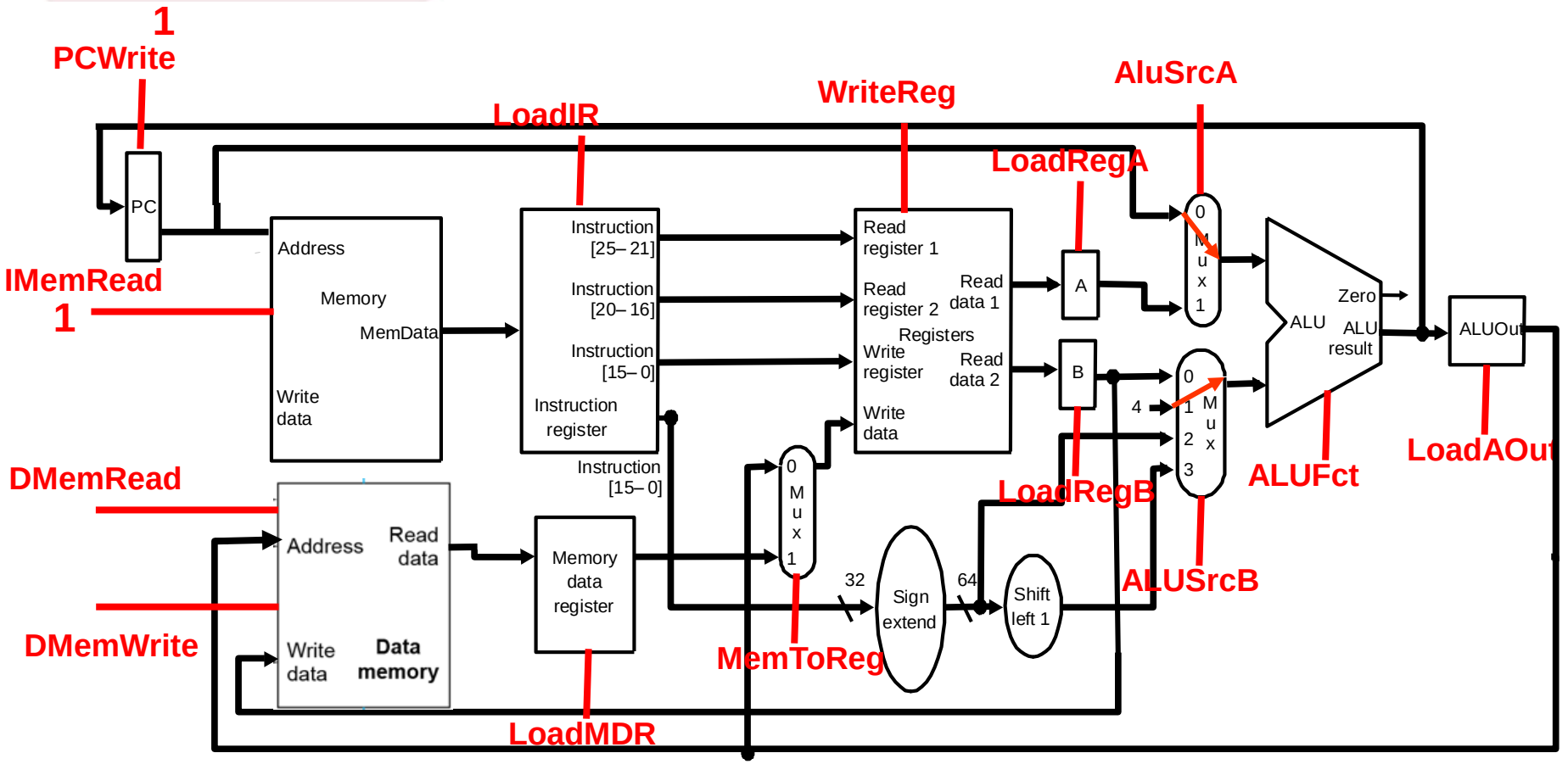
Implementando em ciclos

- Busca de Instrução
  - IR= Memória(PC)
  - PC = PC + 4

Implem

Busca de Instrução

- *IR= Memória(PC), PC = PC + 4*

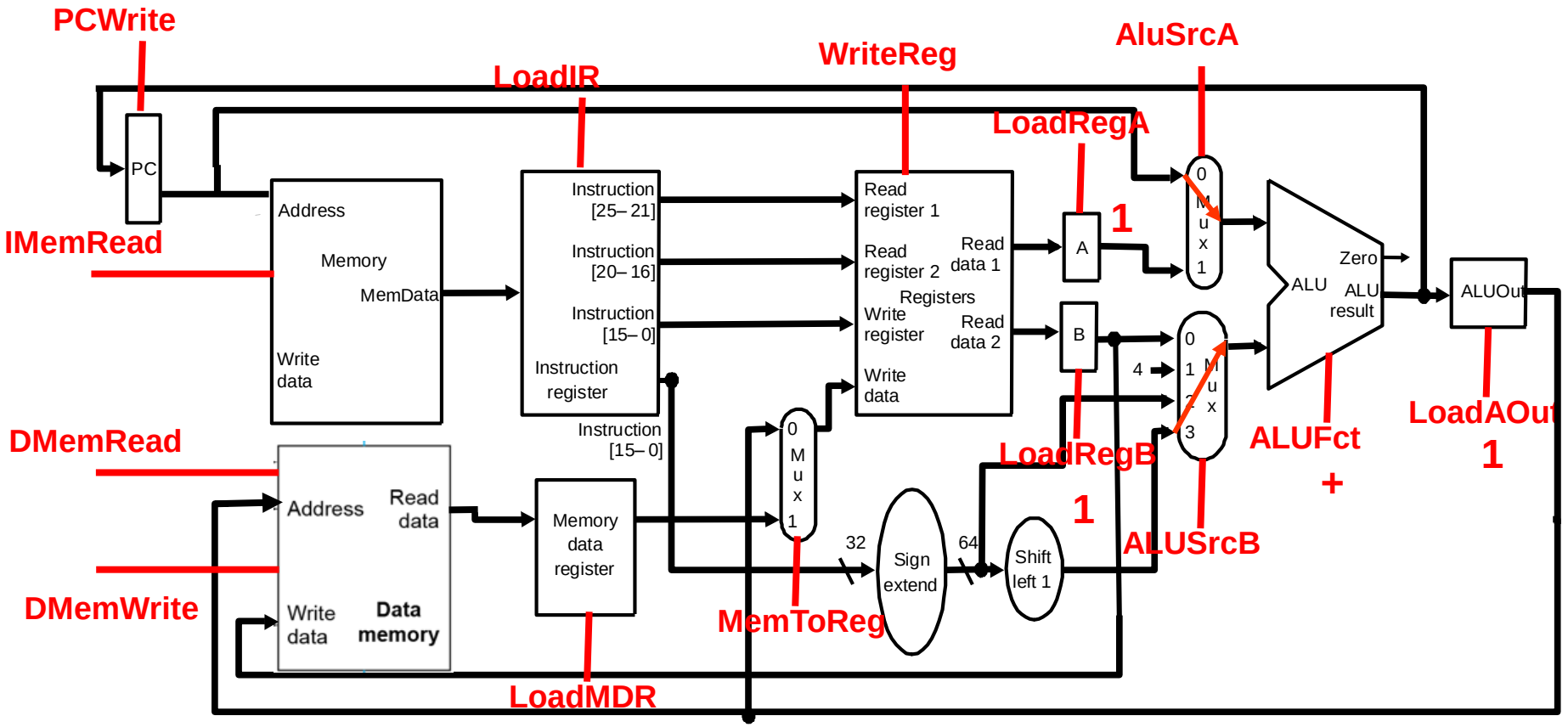


Implementando em ciclos

- Busca de Instrução
  - IR= Memória(PC)
  - PC = PC + 4
- Decodificação e leitura registradores
  - A= Reg(IR(25-21))
  - B= Reg(IR(20-16))
  - ALUout = PC + (desl >>1)

Implem

- *Decodificação e leitura registradores*
  - *A= Reg(IR(25-21)), B= Reg(IR(20-16))*
  - *AluOut=PC+desl>>1*



Implementação Multi-ciclo

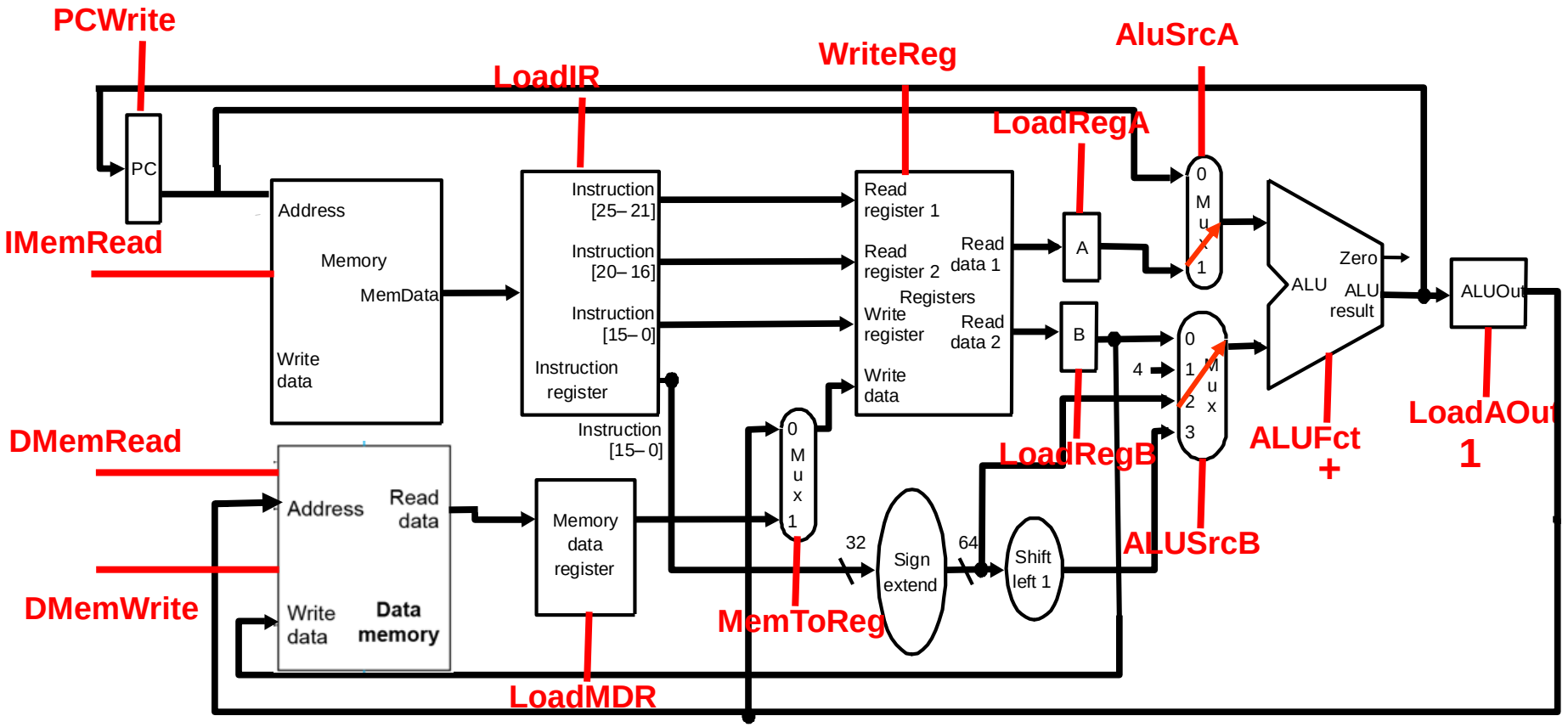
- Load/Store
  - ALUout = A + ext-sinal(IR(32-0))
  - Load: MDR = Memória(ALUout)
  - Reg(dest) = MDR
  - Store: Memória(ALUout) = B

ação Multi-ciclo

Load/Store

–ALUout = A +

ext-sinal(IR(32-0))

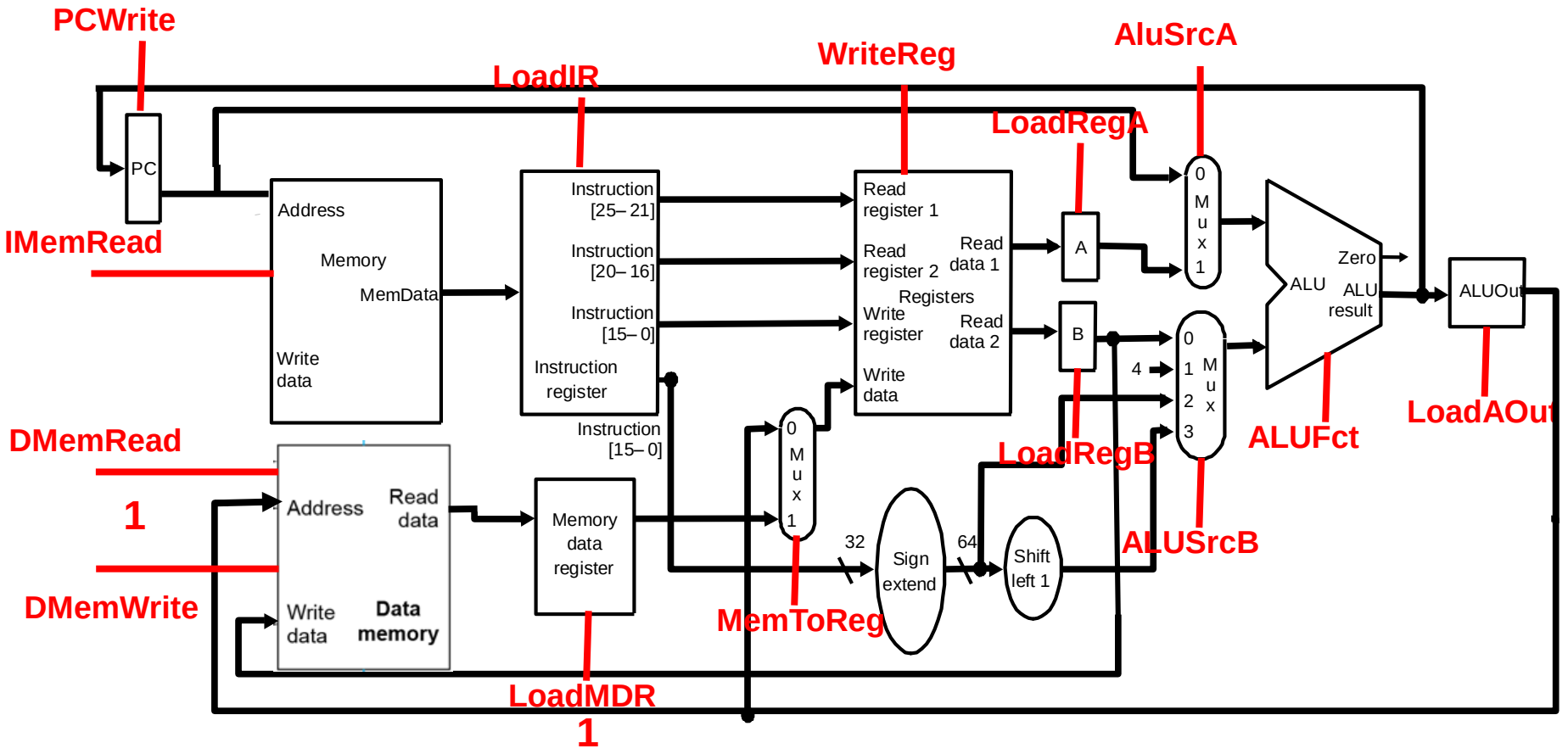


Implementação Multi-ciclo

- Load/Store
  - ALUout = A + ext-sinal(IR(15-0))
  - Load: MDR = Memória(ALUout)
  - Reg(IR(20-16)) = MDR
  - Store: Memória(ALUout) = B

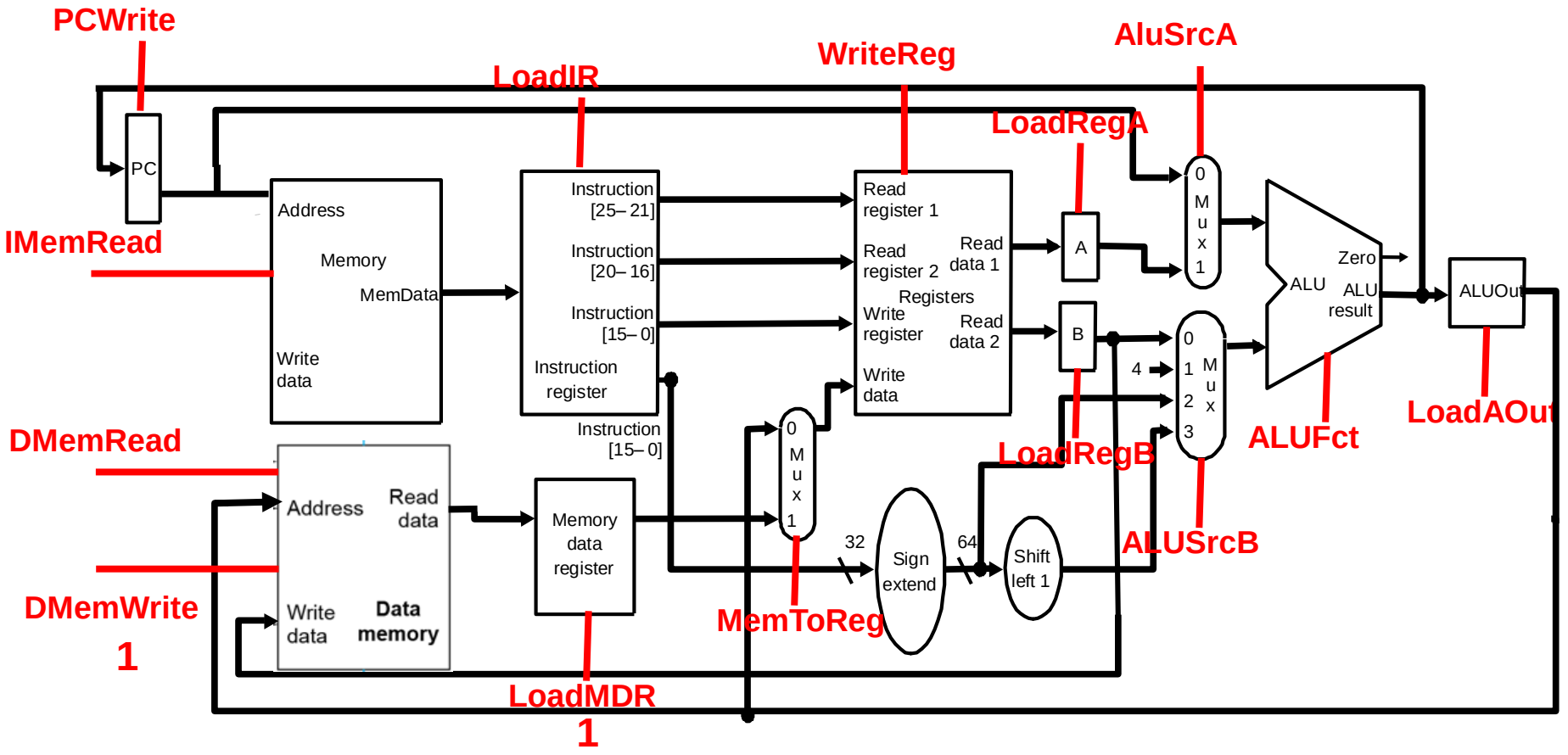
Implementaç

- *Load/Store*
  - *–Load: MDR = Memória(ALUout)*
  - *–Store: Memória(ALUout) = B*



Implementaç

- *Load/Store*
  - *–Load: MDR = Memória(ALUout)*
  - *–Store: Memória(ALUout) = B*

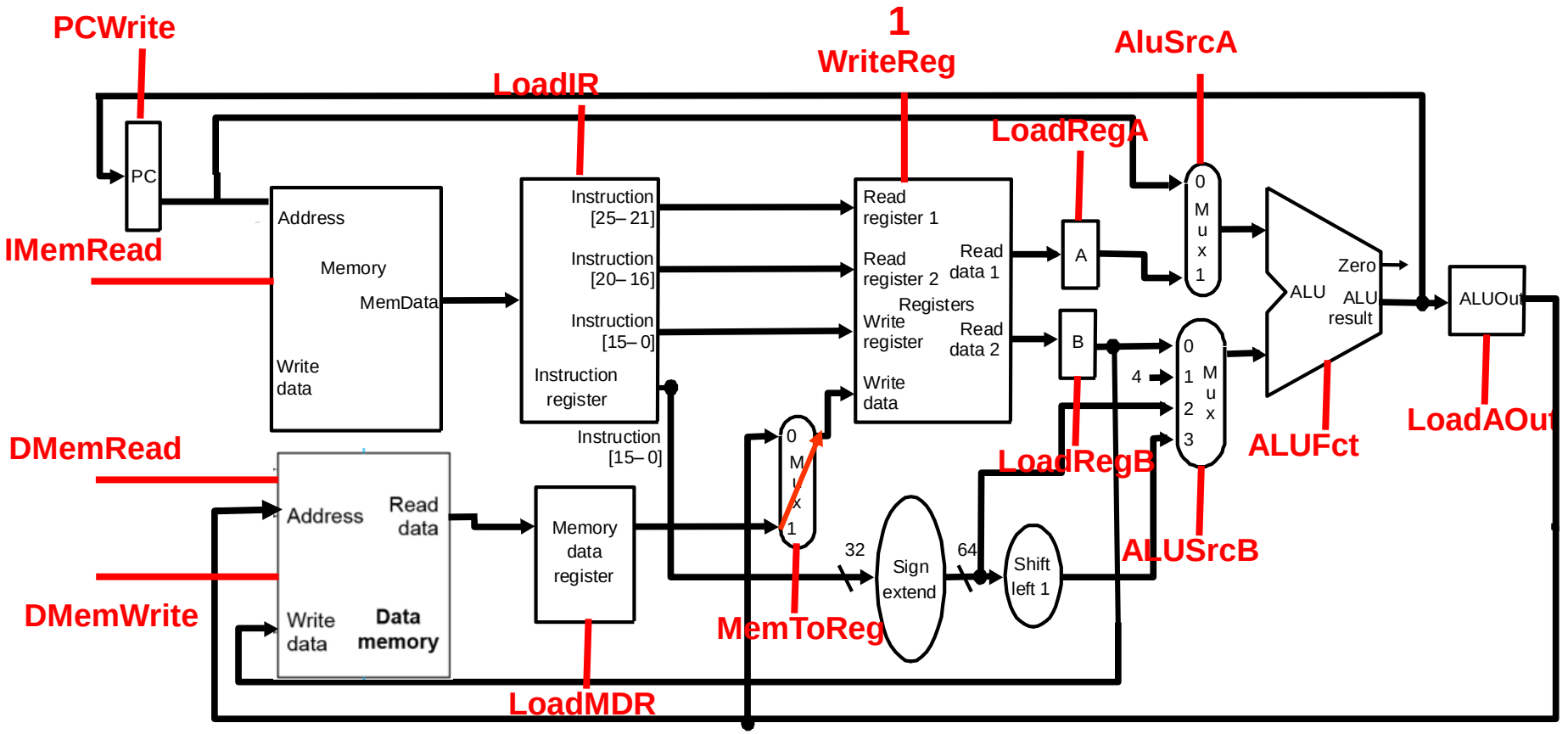


Implementação Multi-ciclo

- Load/Store
  - ALUout = A + ext-sinal(IR(15-0))
  - Load: MDR = Memória(ALUout)
  - Reg(IR(20-16)) = MDR
  - Store: Memória(ALUout) = B

Implementação Multi-ciclo

- *Load*
  - *–Reg(IR(20-16)) = MDR*

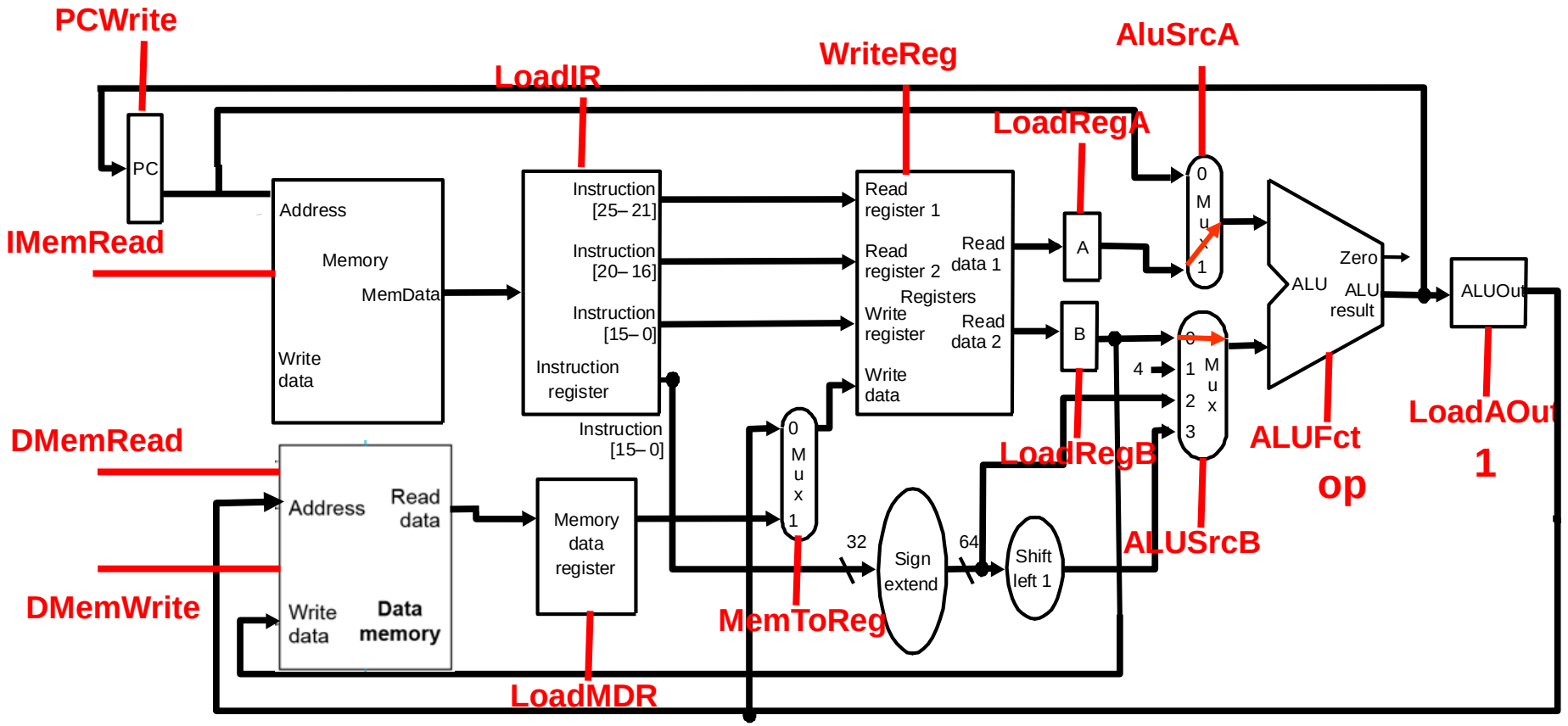


Implementação Multi-ciclo

- Aritmética
  - ALUout = A op B
  - Reg(IR(15-11)) = ALUout

ação Multi-ciclo

- *Aritmética*
  - *–ALUout = A op B*

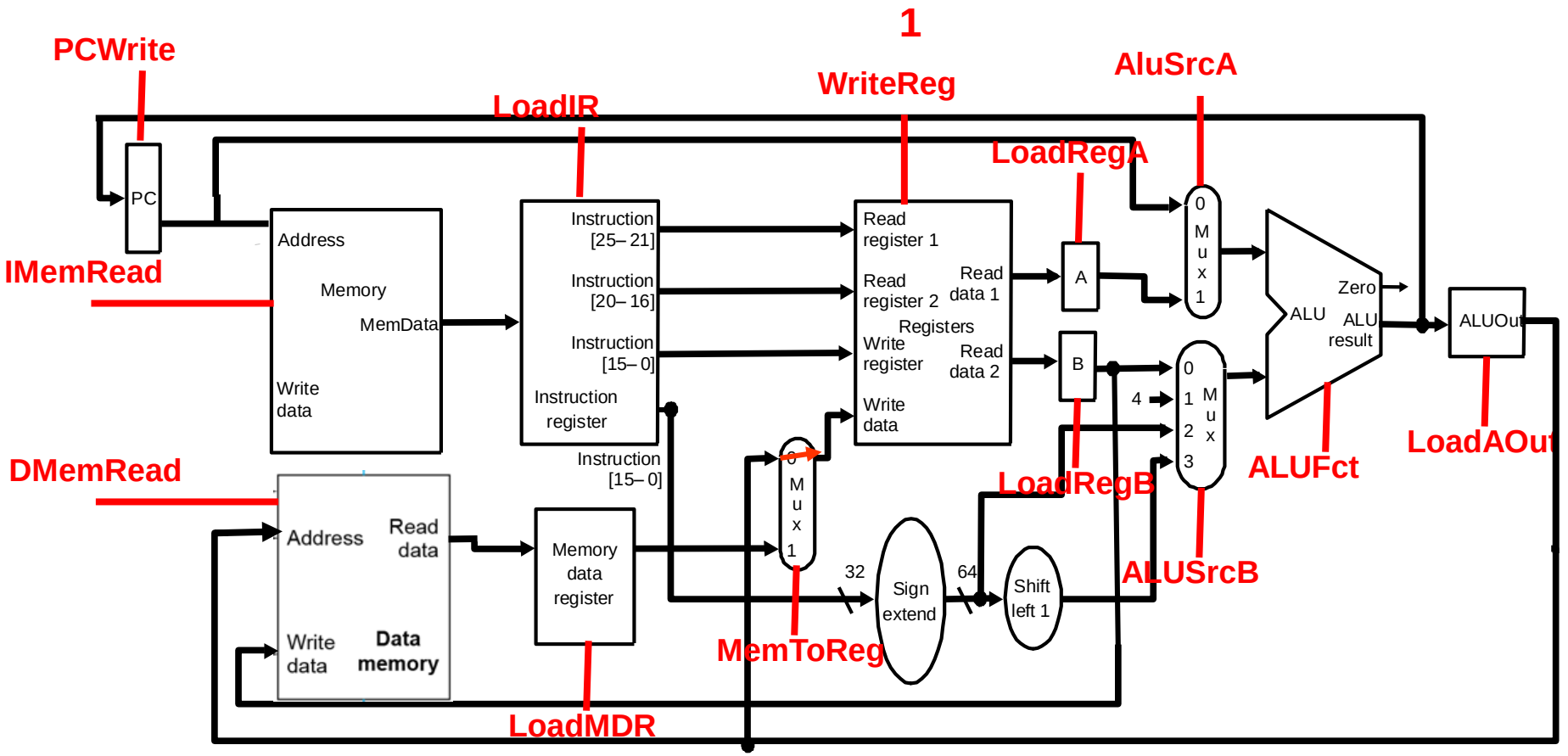


Implementação Multi-ciclo

- Aritmética
  - ALUout = A op B
  - Reg(IR(15-11)) = ALUout

Implementaç

- *Aritmética*
  - *–Reg(IR(15-11)) = ALUout*

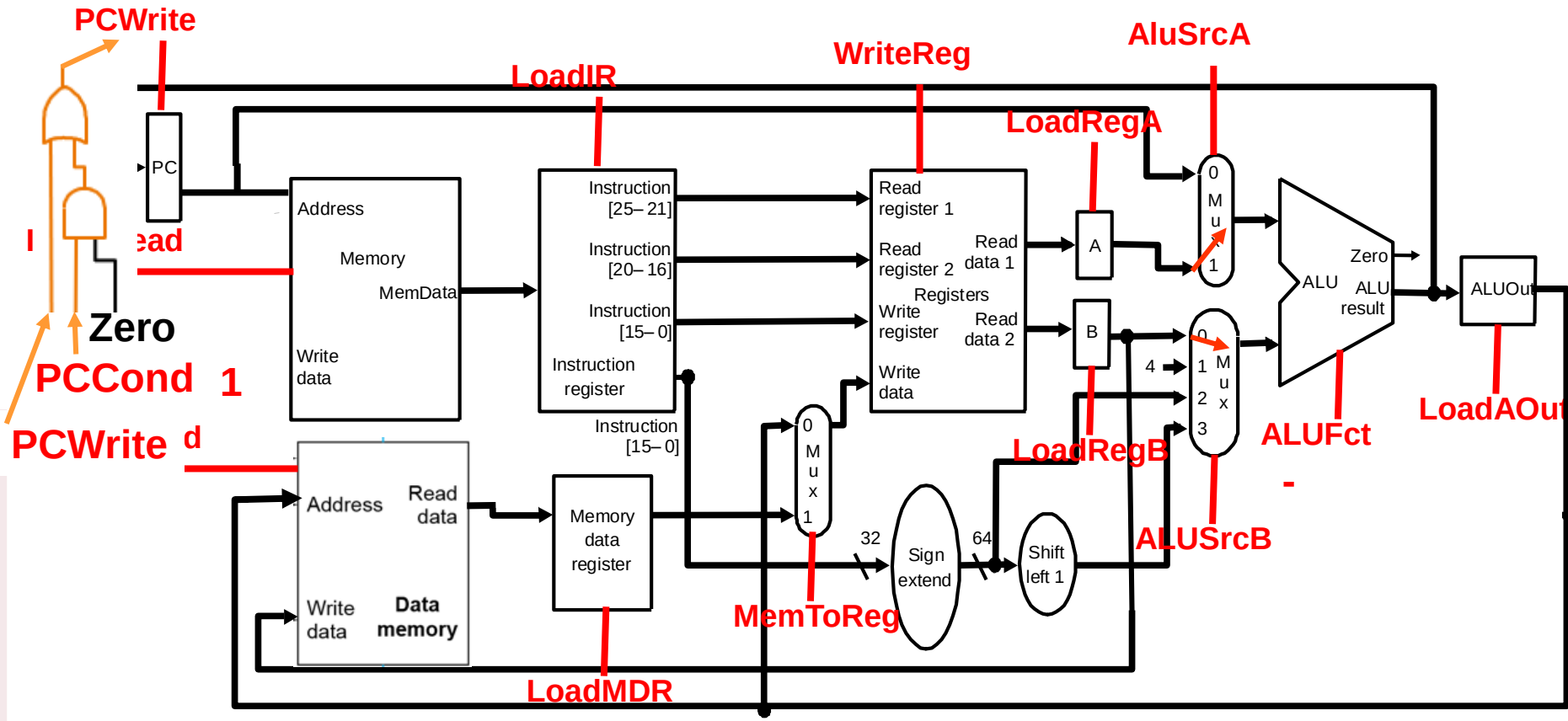


Implementação Multi-ciclo

- Branch
  - if (A ==B) PC = ALUout

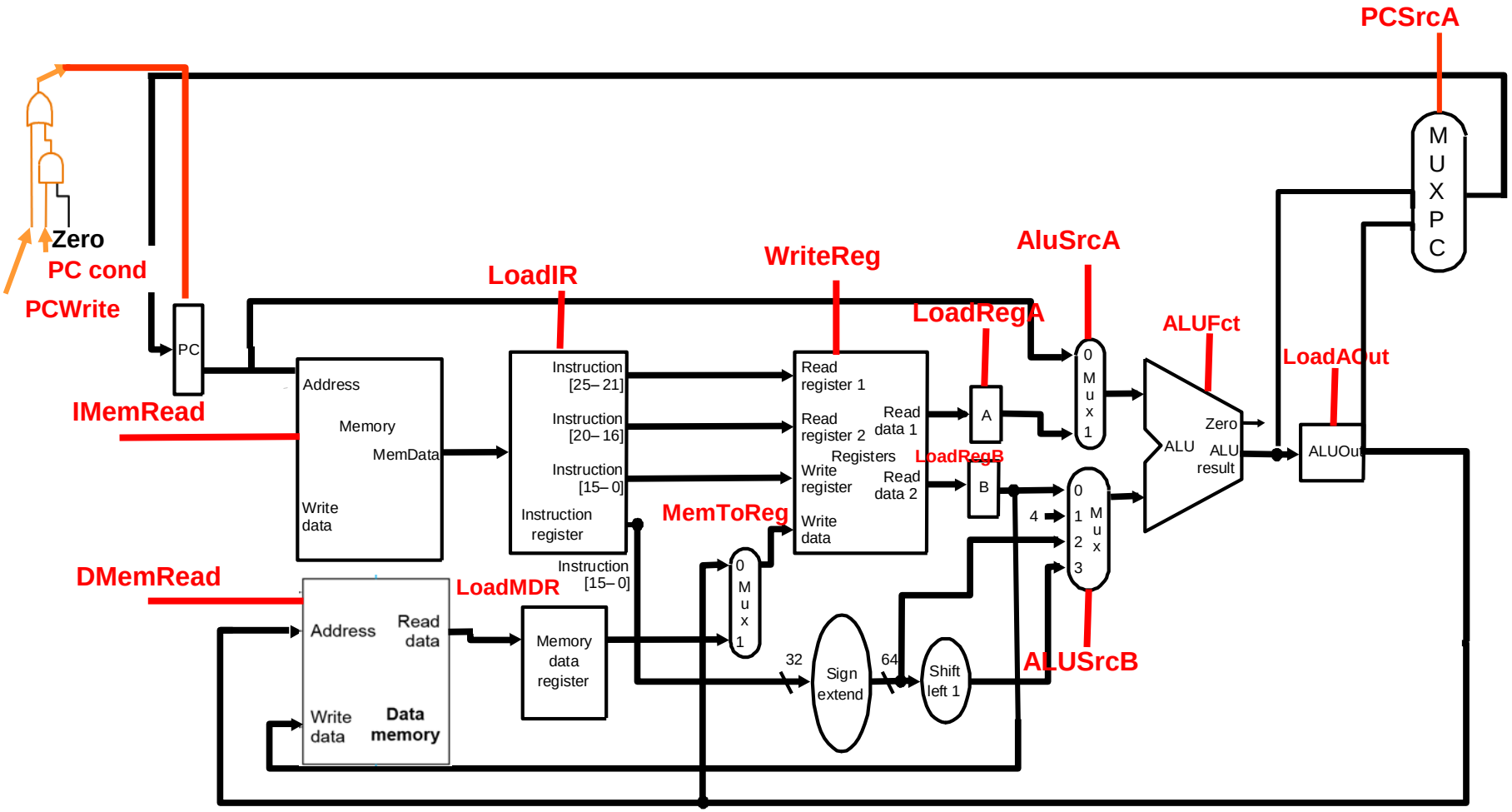
ação Multi-ciclo

- *Branch*
  - *if (A ==B) PC = ALUout*

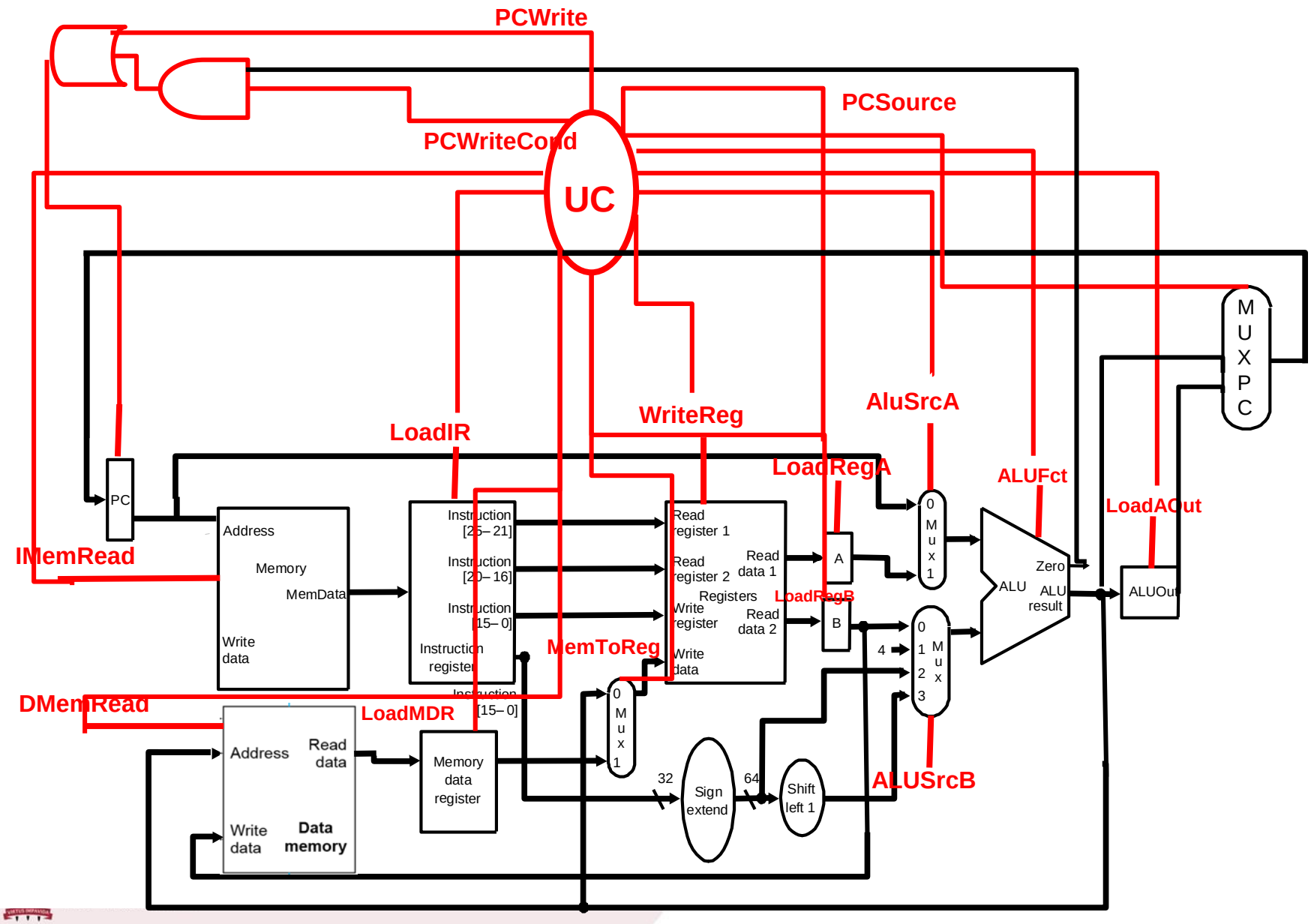


ação Multi-ciclo

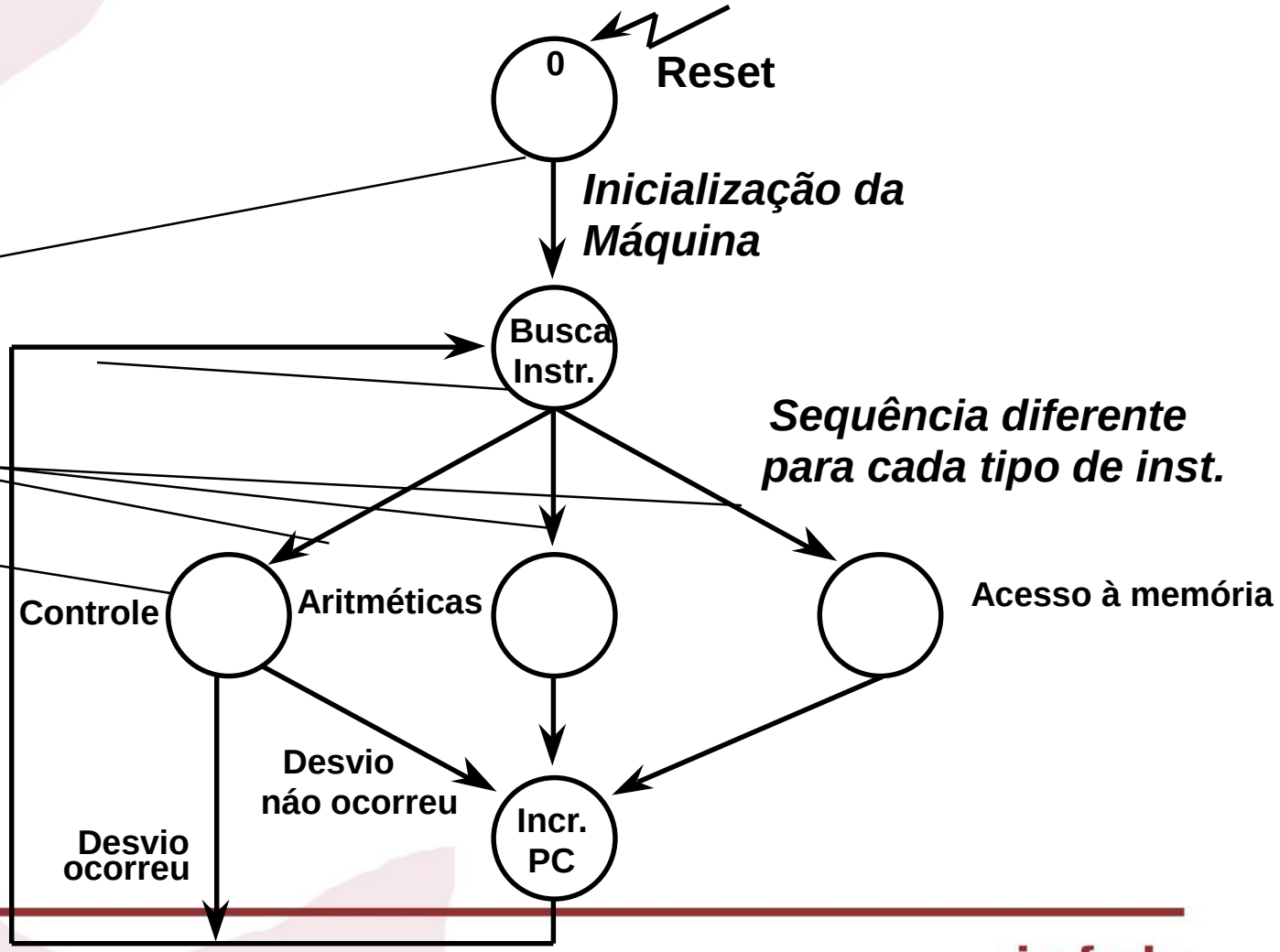
- *Branch*
  - *if (A ==B) PC = ALUout*



Implementação Multi-ciclo



Unid. Controle - Diagr. Estados

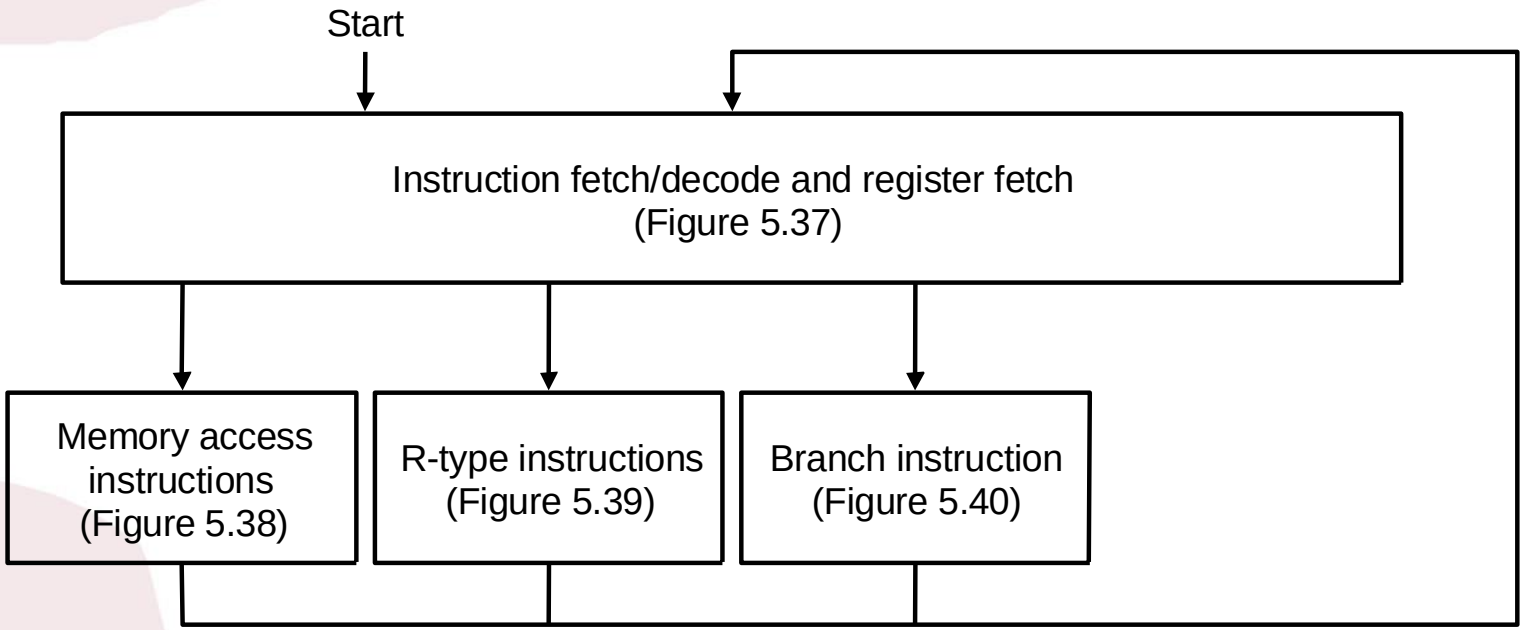


- Reset
- Busca
- Decodif.
- Exec.

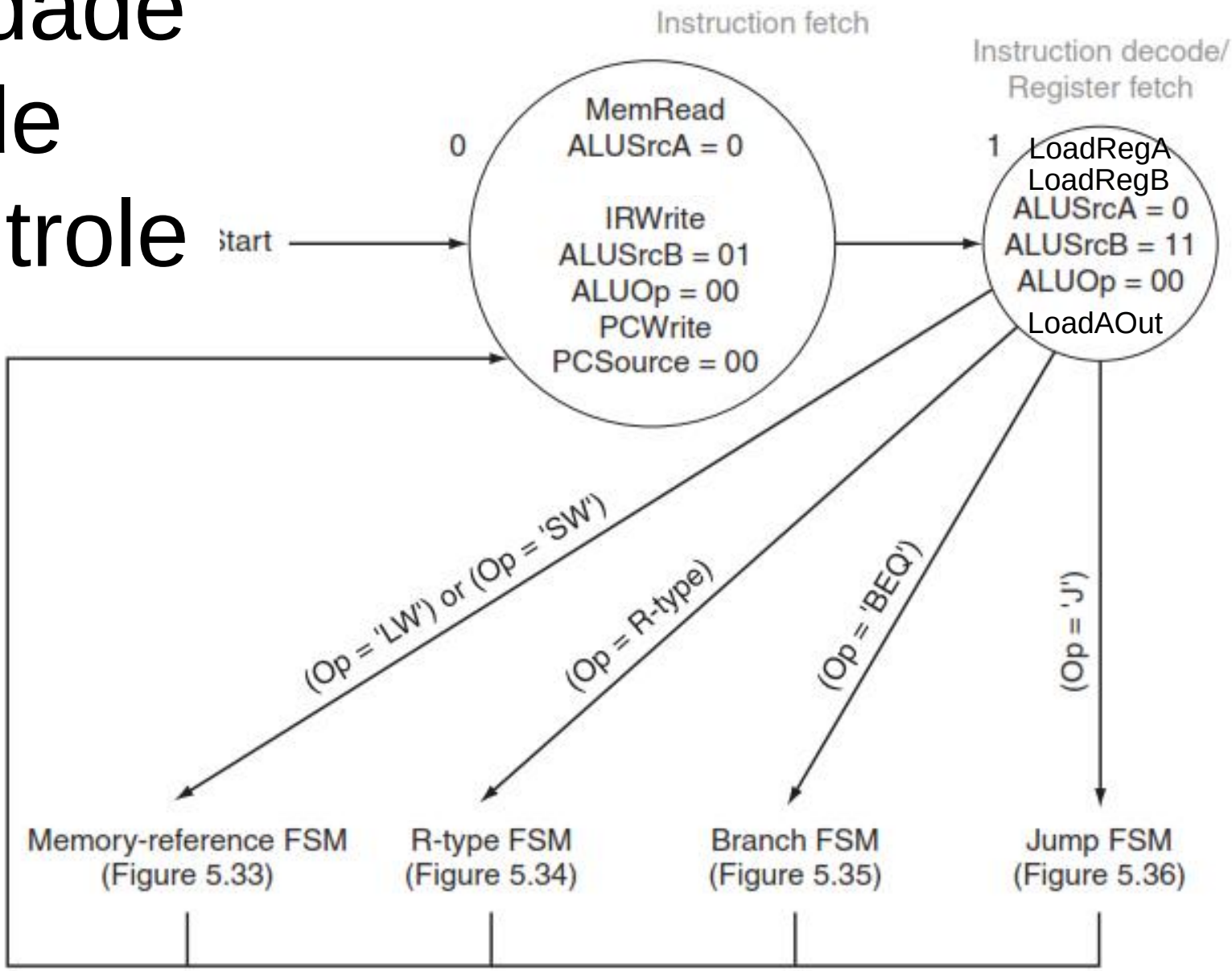
Três classes de instruções:

- Aritméticas
- Movimentação
- Controle

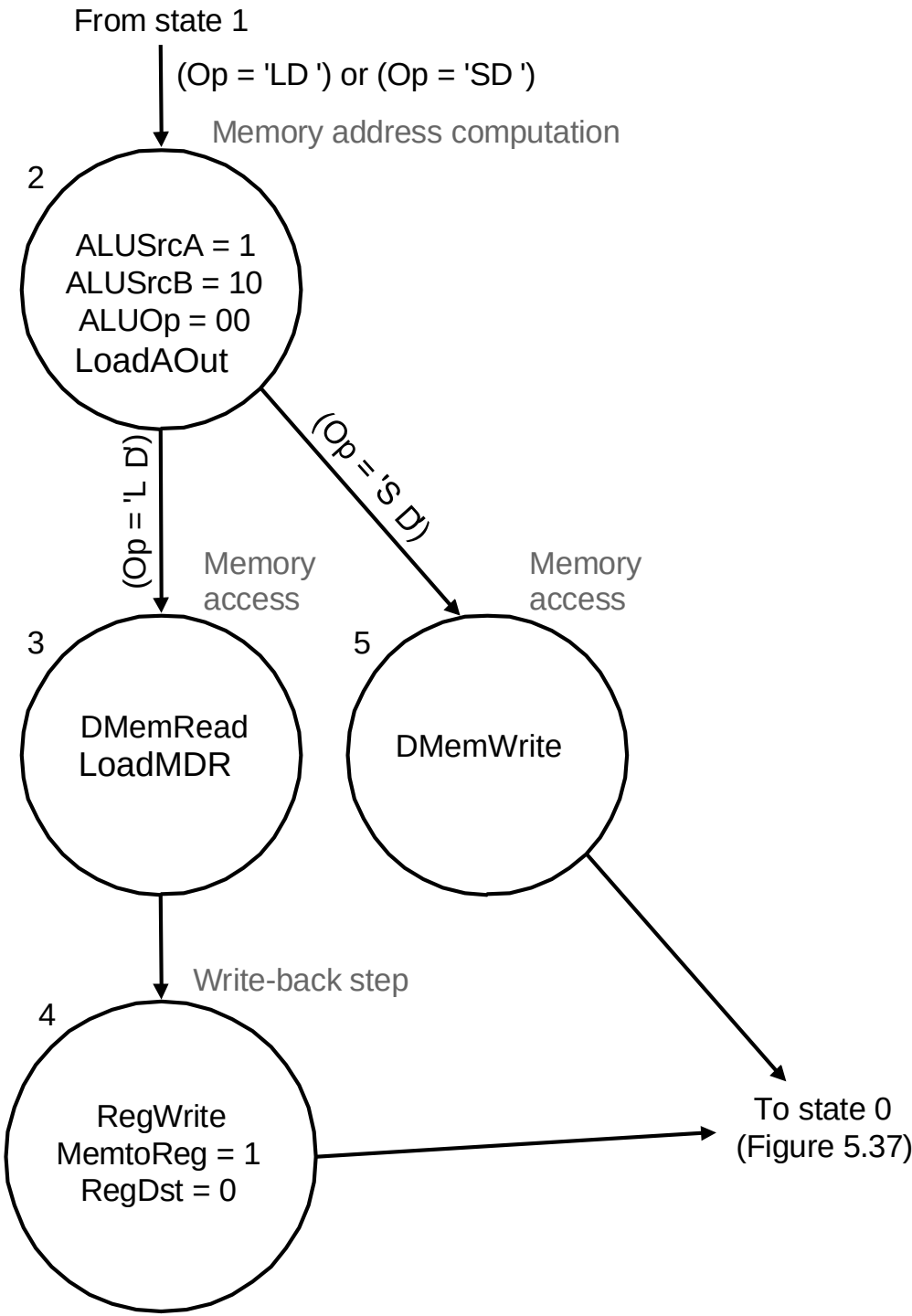
Unidade de Controle



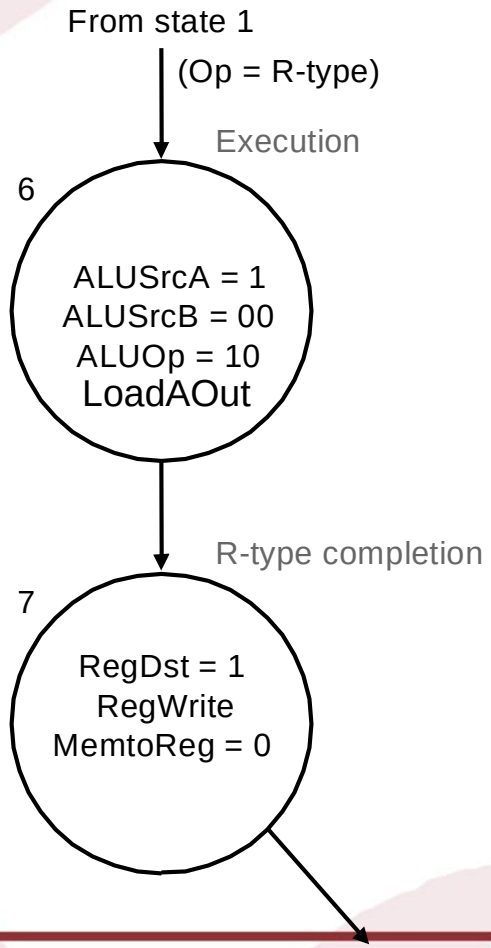
Unidade de Controle



Unidade de Controle

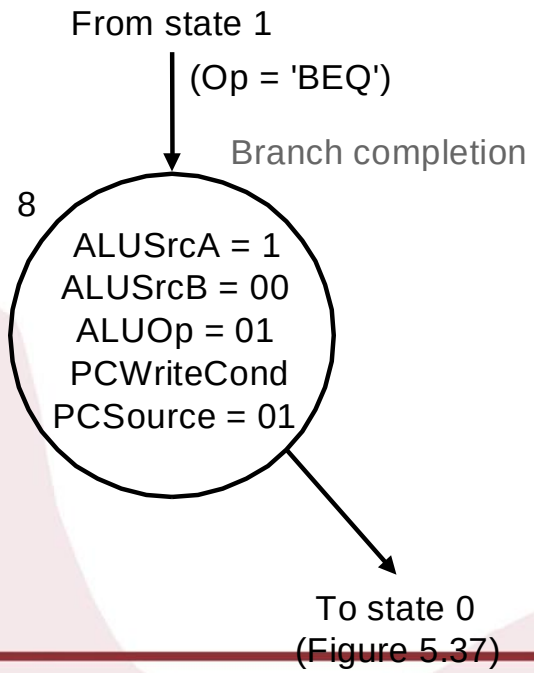


Aritméticas

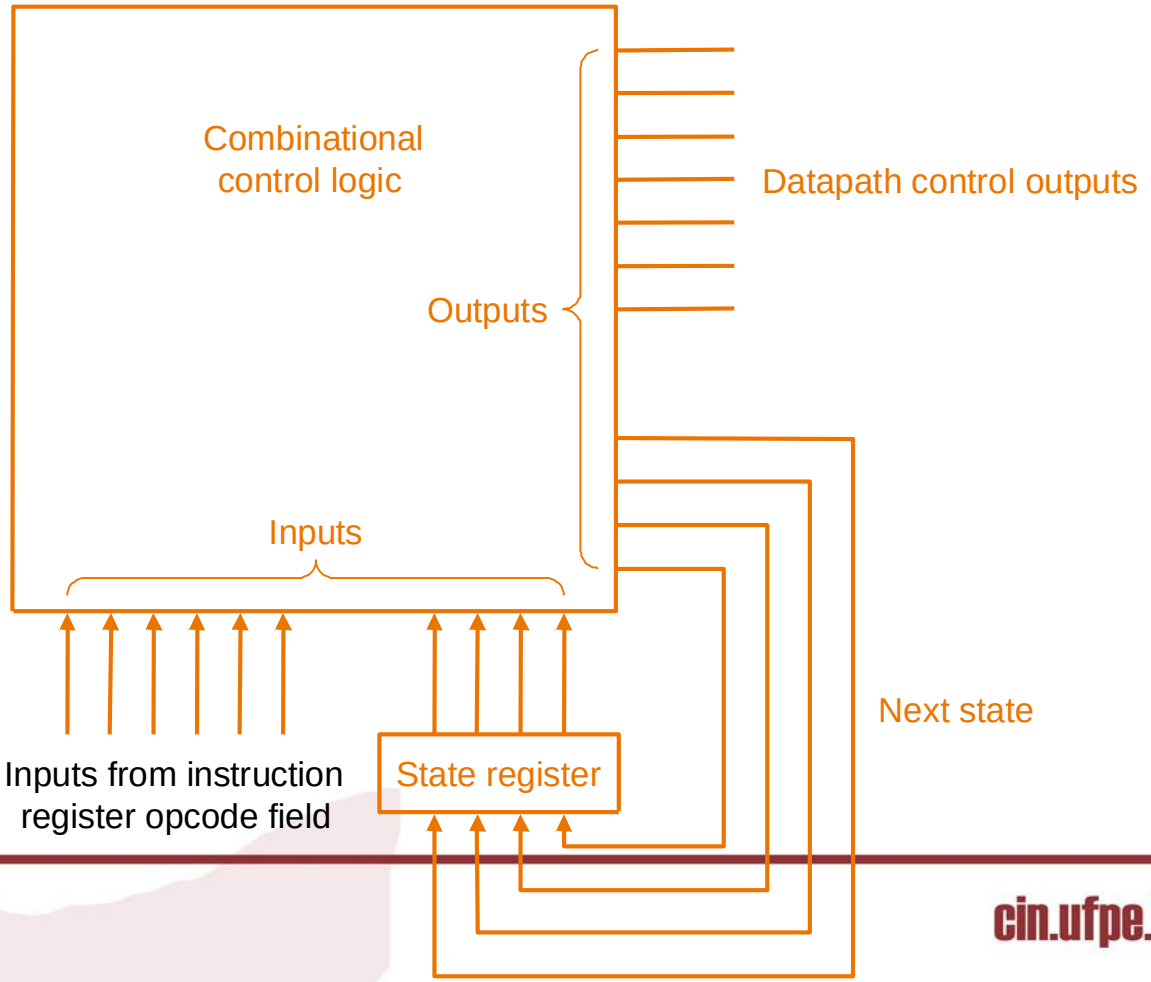


Desvio

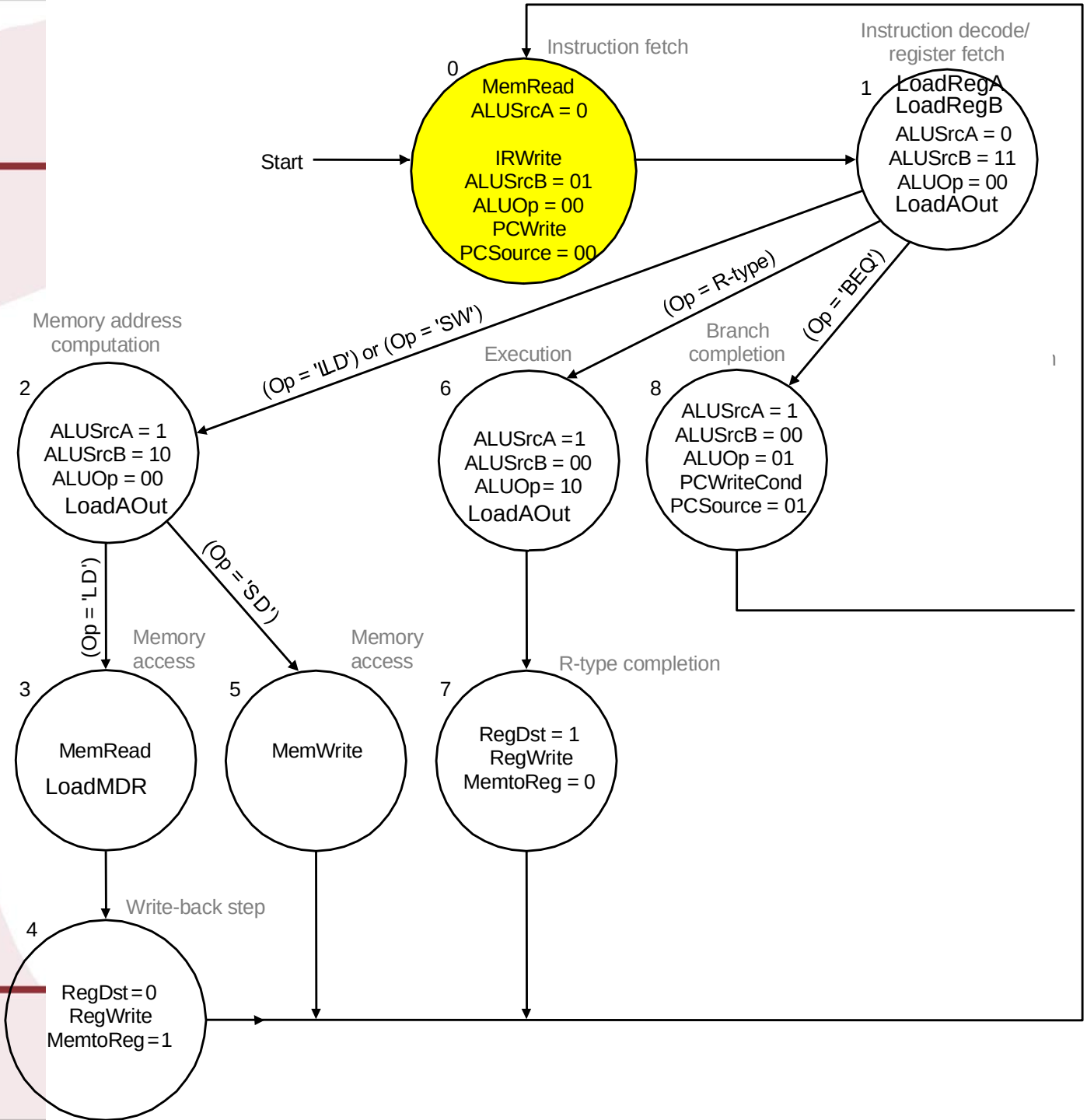
- BEQ


From state 1
(Op = 'BEQ')
Branch completion
8
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCWriteCond
PCSource = 01
To state 0
(Figure 5.37)

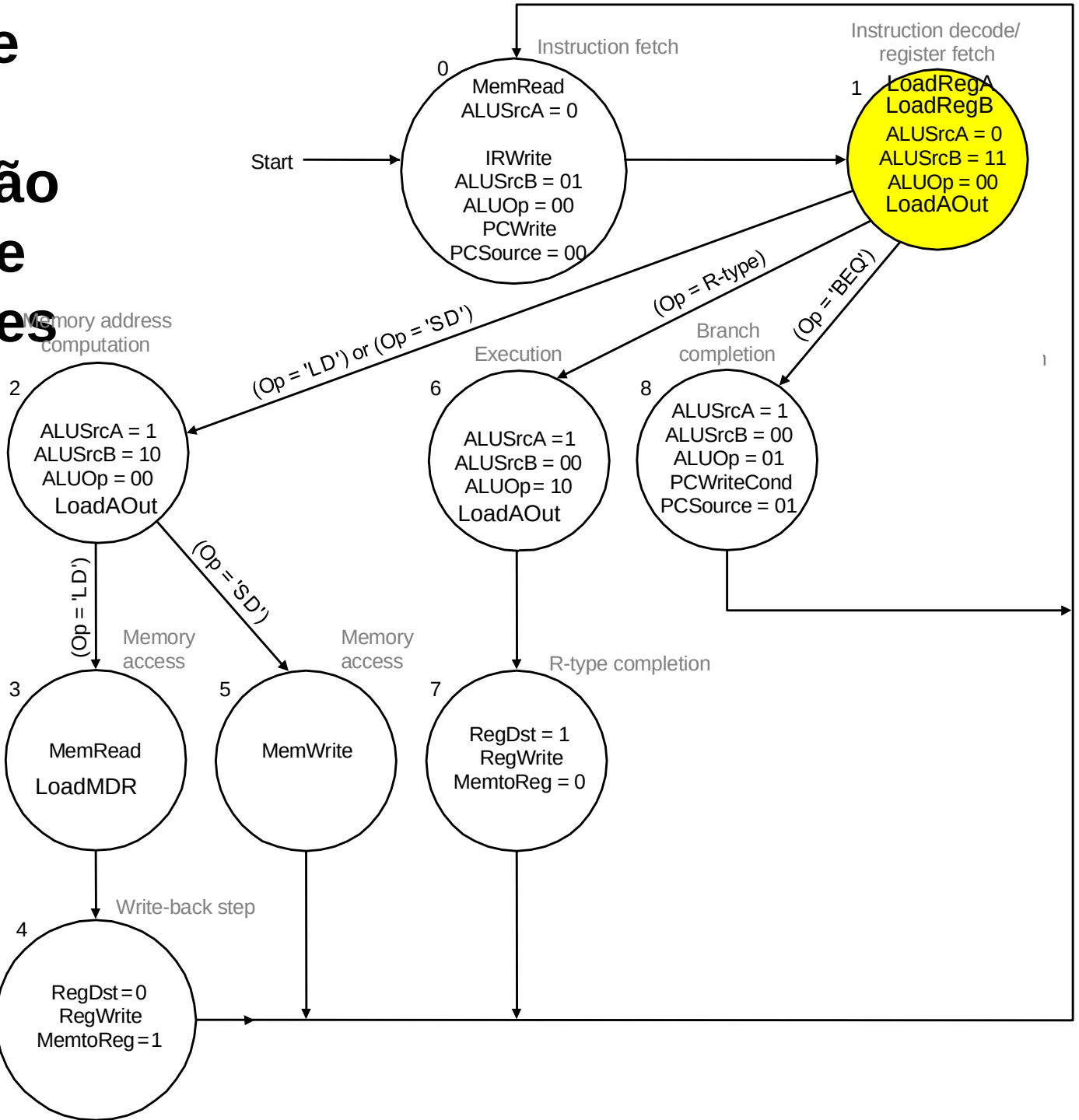
Unidade de Controle - FSM



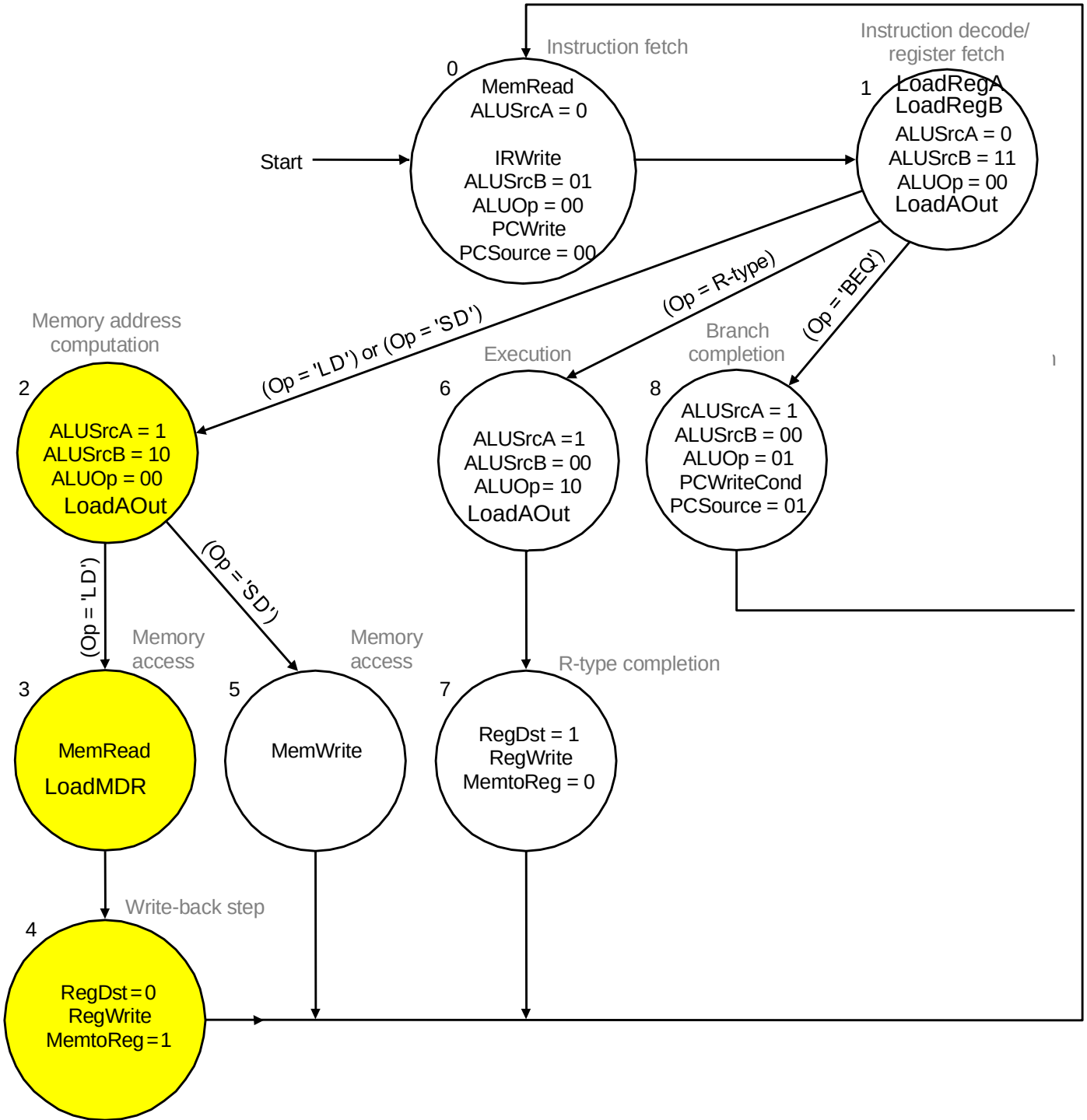
Unidade de Controle: Busca de Instrução



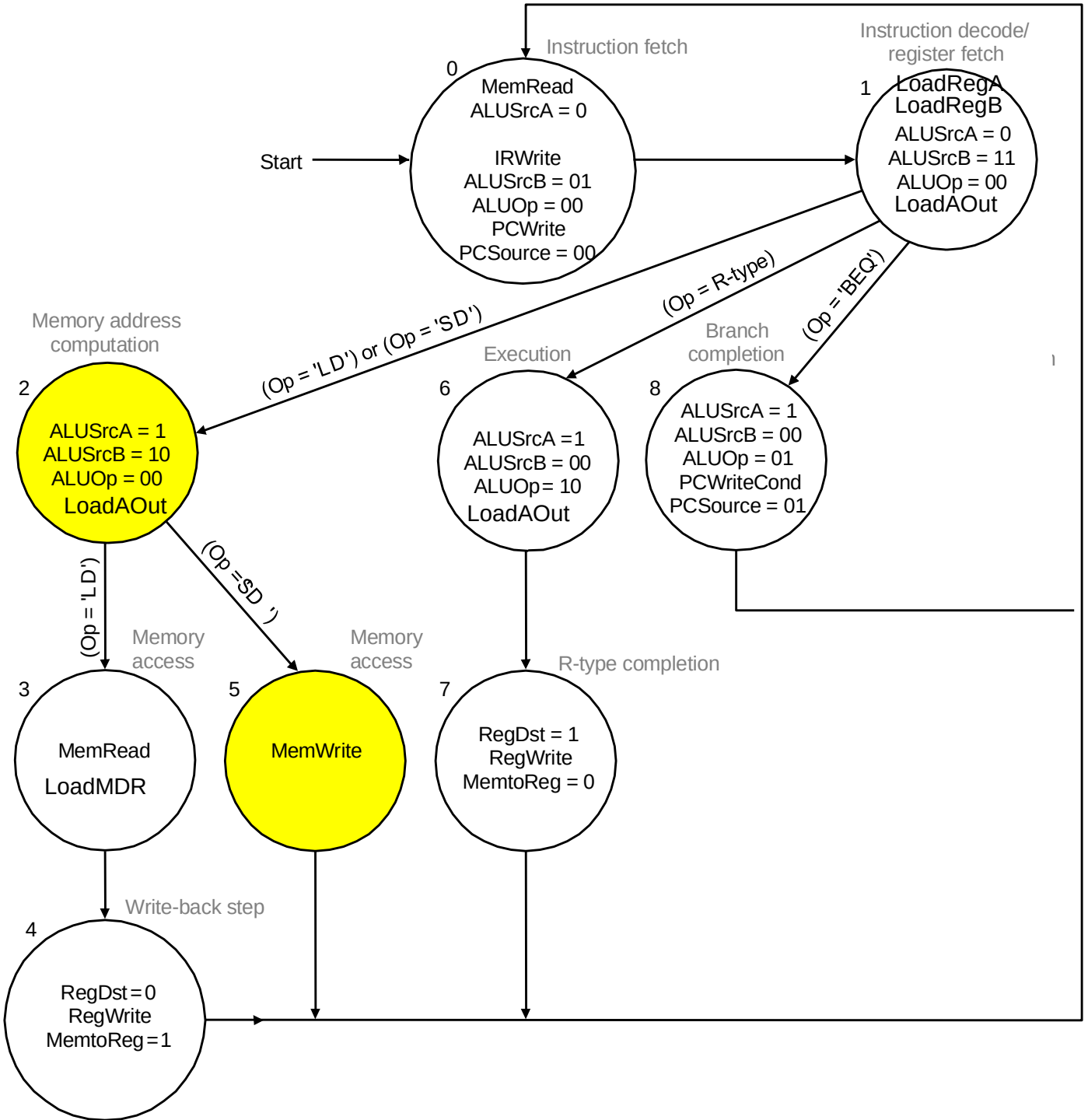
Unidade de Controle: Decodificação e Leitura de Registradores



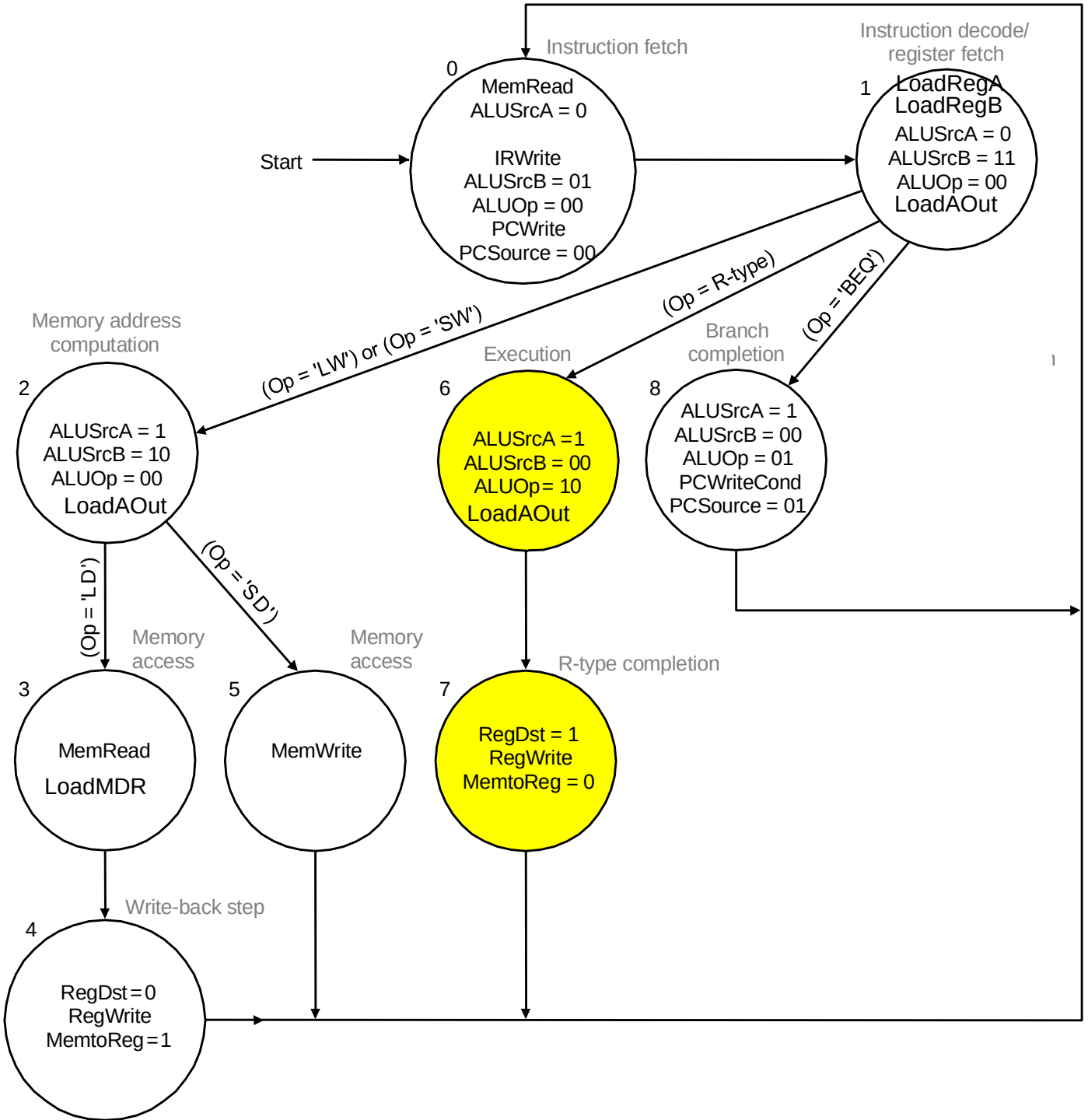
Unidade de Controle: Execução do LD



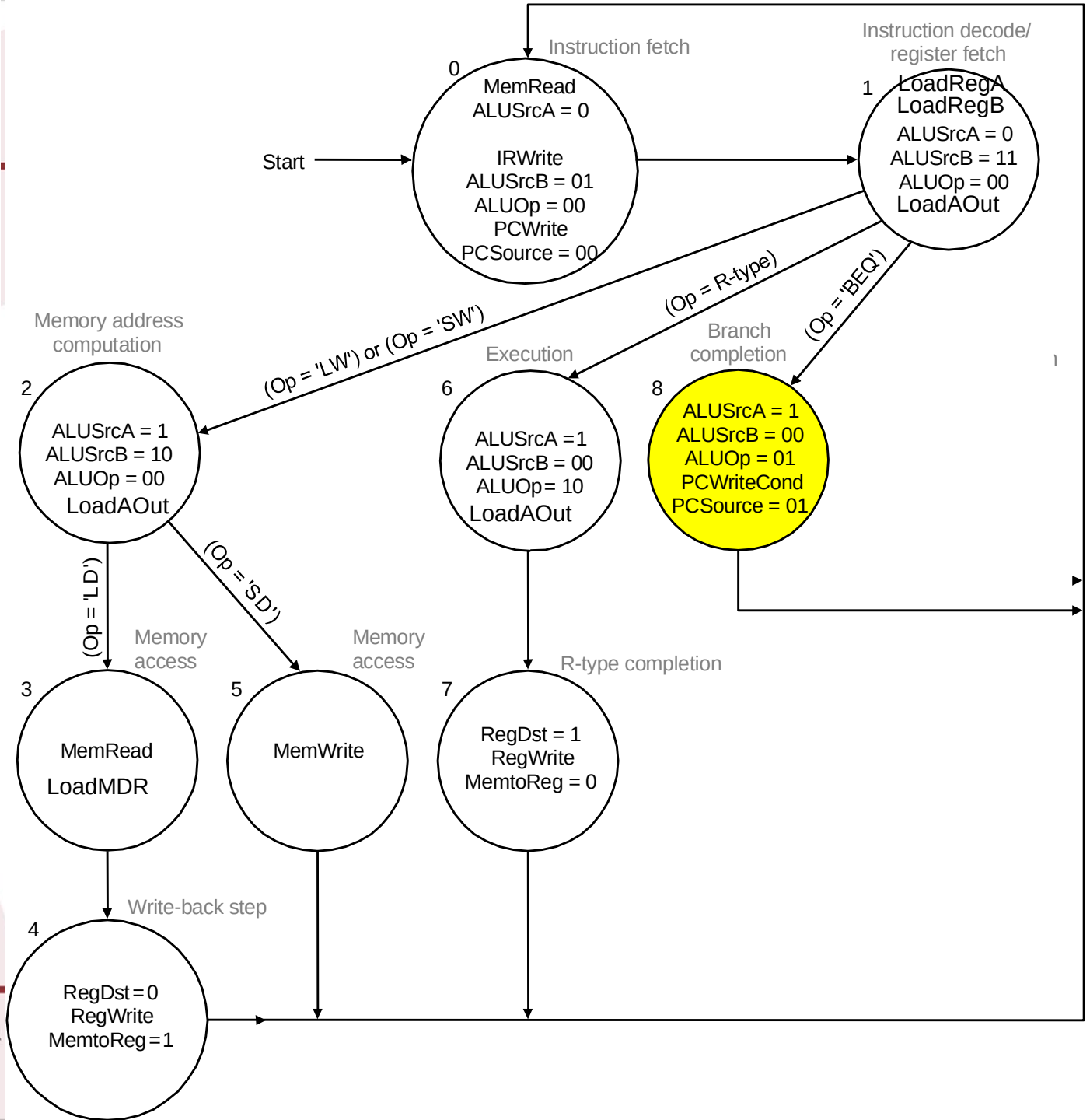
Unidade de Controle: Execução do SD



Unidade de Controle: Execução de ADD



Unidade de Controle: Execução do BEQ



Centro de Informática U·F·P·E

CPU: Exceções

Exceções

- Sequência de execução é alterada devido a eventos não esperados:
  - internos:
    - opcode inexistente
    - overflow
    - divisão por zero
  - externos:
    - dispositivo de entrada/saída

Exceções

- Execução do programa é interrompida e uma rotina de tratamento é executada
  - Valor do PC deve ser guardado
  - O endereço da rotina de tratamento deve ser carregado em PC

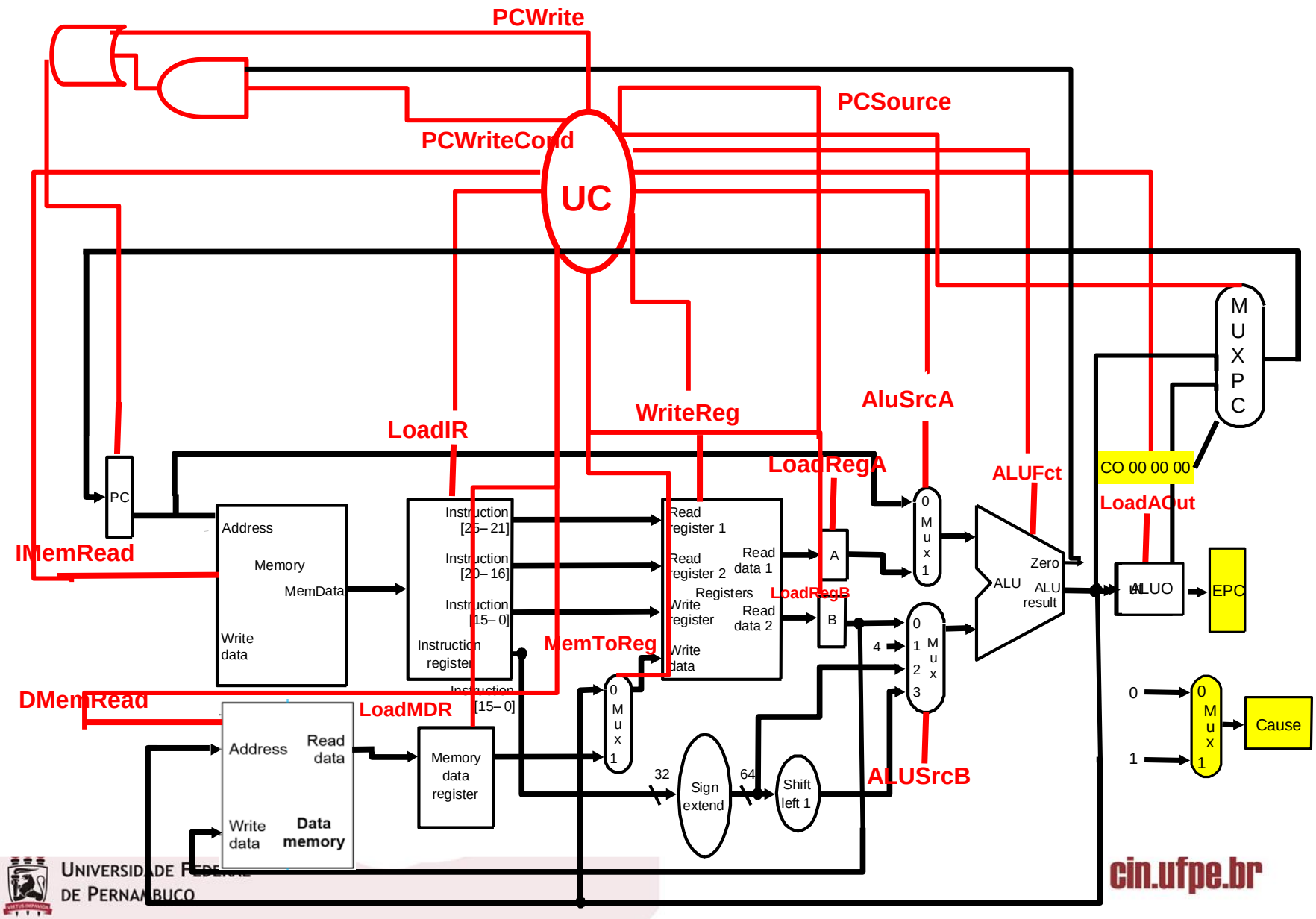
Exceções

- Onde guardar o endereço do PC
  - Registrador
    - RISC V: EPC
  - Pilha
- Onde o endereço da sua subrotina deve ser guardado
  - Valor fixo
    - precisa-se saber a causa da exceção
  - Vetor de endereços na memória

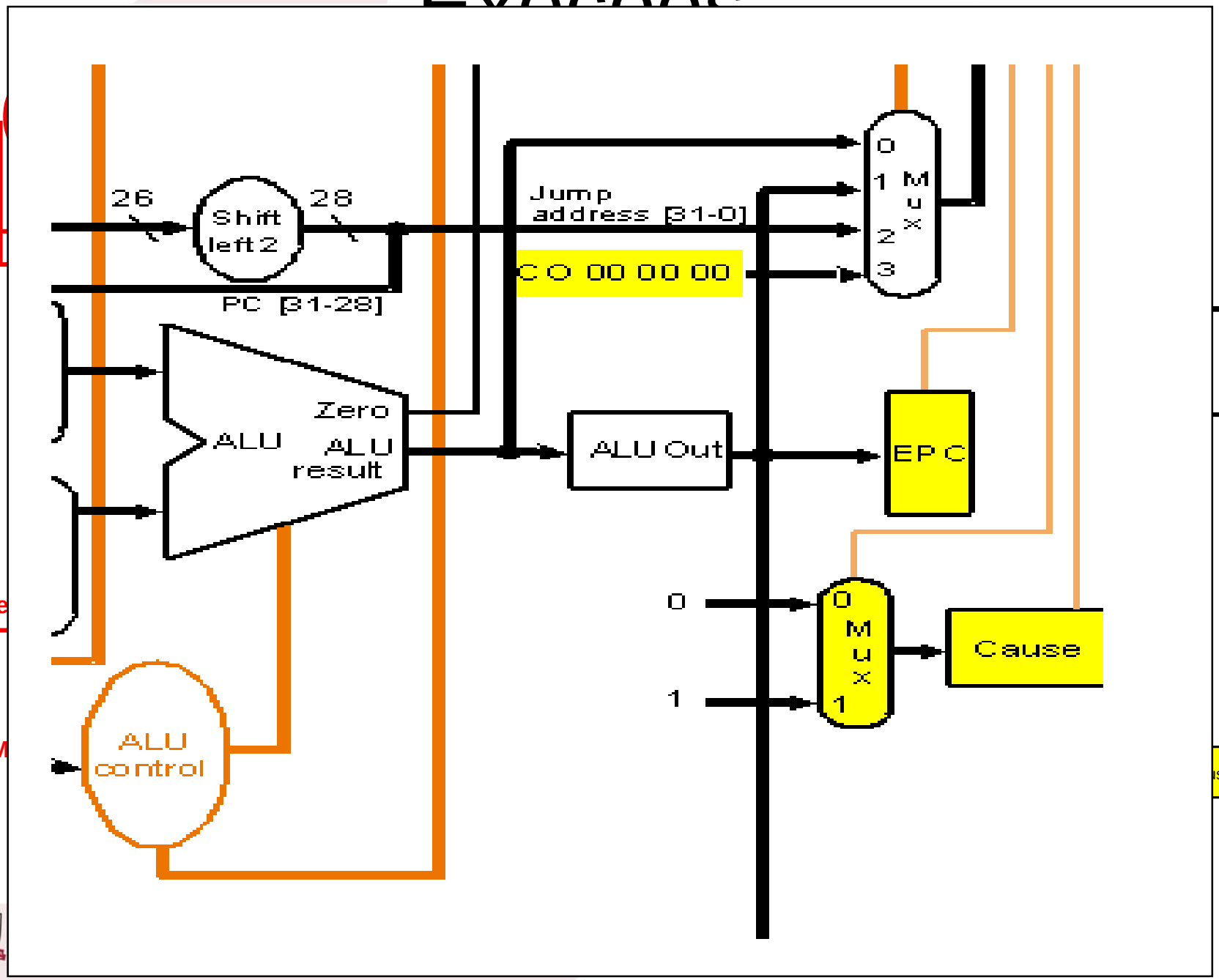
Exceções - MIPS

- Tipos de Exceções:
  - Instrução Indefinida
    - endereço: C0 00 00 00
  - Overflow aritmético
    - endereço: C0 00 00 20
- Registrador EPC
  - guarda endereço da instrução afetada
- Registrador de Causa
  - identifica o tipo de evento que causou a exceção

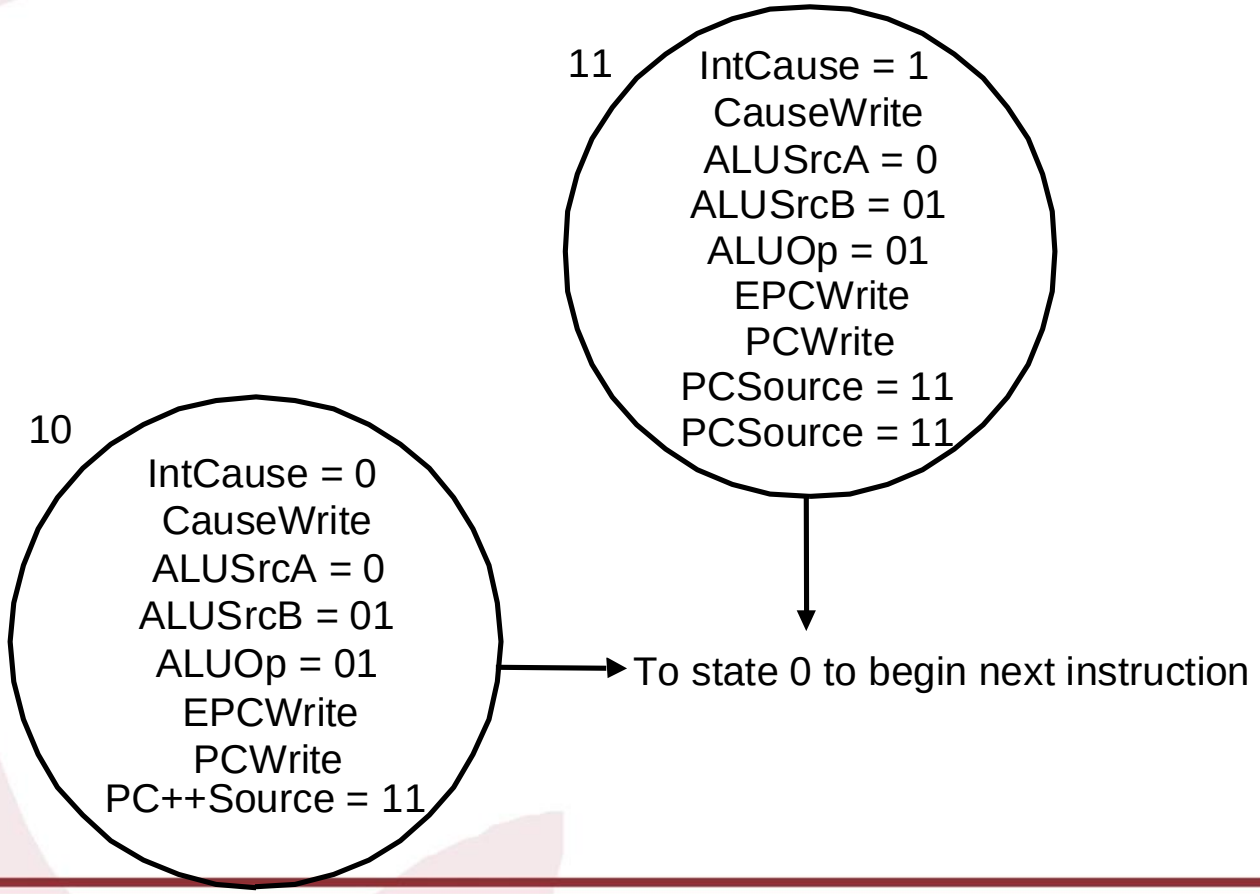
Exeções


PCWrite
PCSource
PCWriteCond
UC
LoadIR
WriteReg
AluSrcA
LoadRegA
ALUFct
MUXPC
CO 00 00 00
LoadAOut
IMemRead
PC
Address
Memory
MemData
Write data
Instruction [25–21]
Instruction [20–16]
Instruction [15–0]
Instruction register
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
A
B
LoadRegB
0 Mux 1
ALU
Zero
ALU result
ALUO
EPC
4
0 1 2 3 Mux
MemToReg
DMemRead
LoadMDR
Address
Read data
Write data
Data memory
Memory data register
0 Mux 1
32
Sign extend
64
Shift left 1
ALUSrcB
0
1
0 Mux 1
Cause

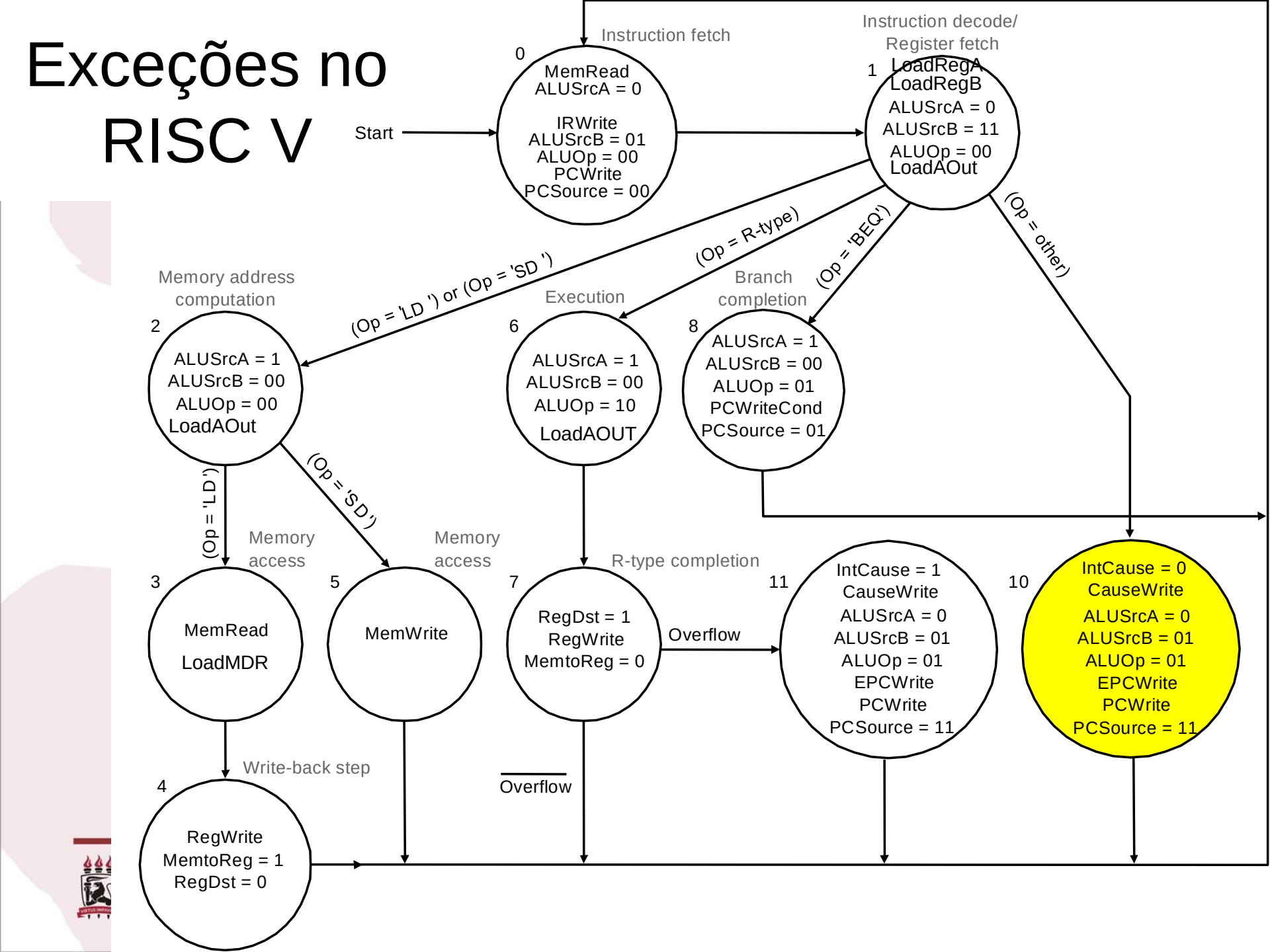
Exceções


26
Shift left 2
28
Jump address [31-0]
PC [31-28]
C O 00 00 00
0
1
2
3
M u x
ALU
Zero
ALU result
ALU Out
EP C
0
1
M u x
Cause
ALU control

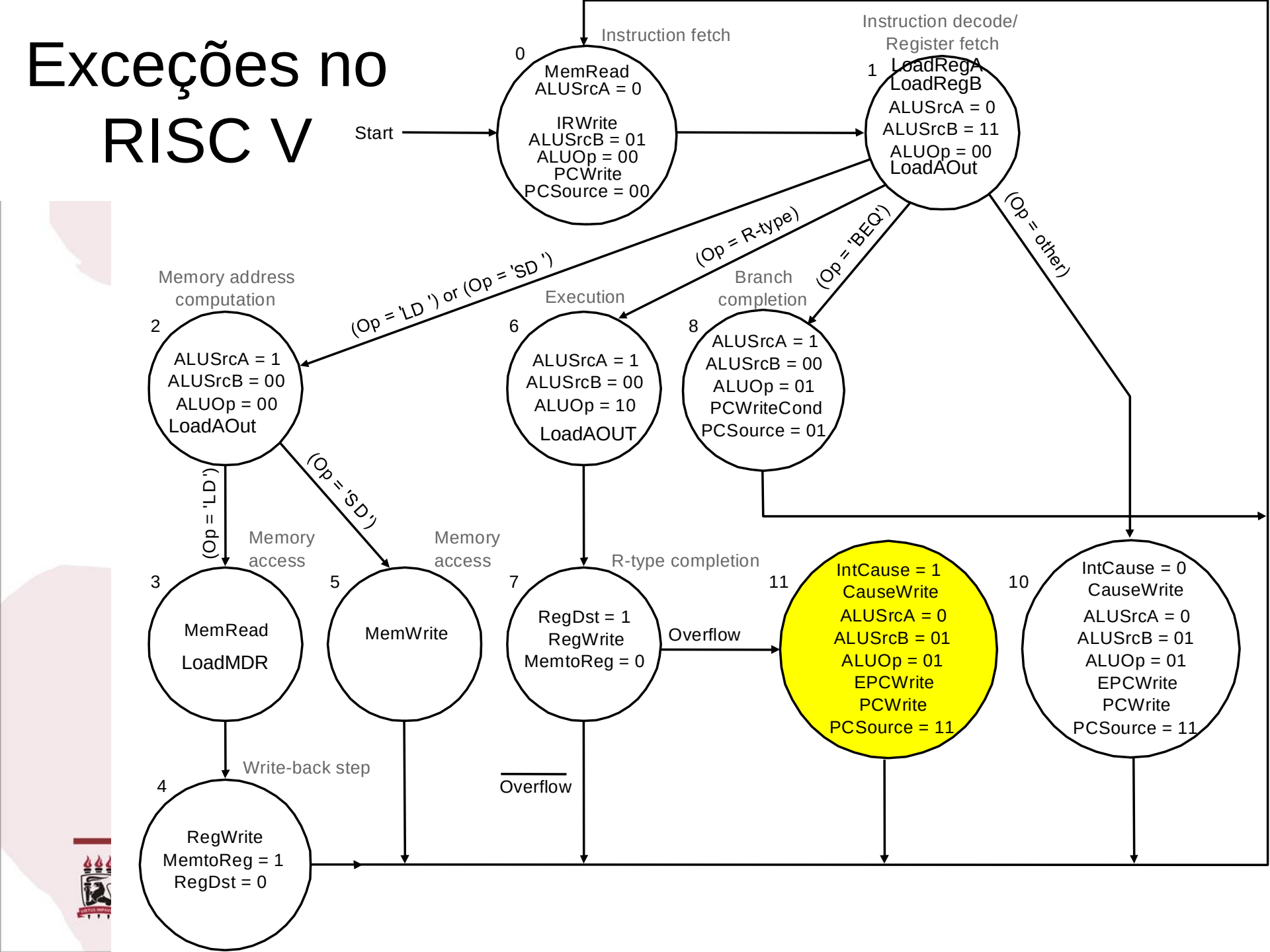
Exceções no RISC V



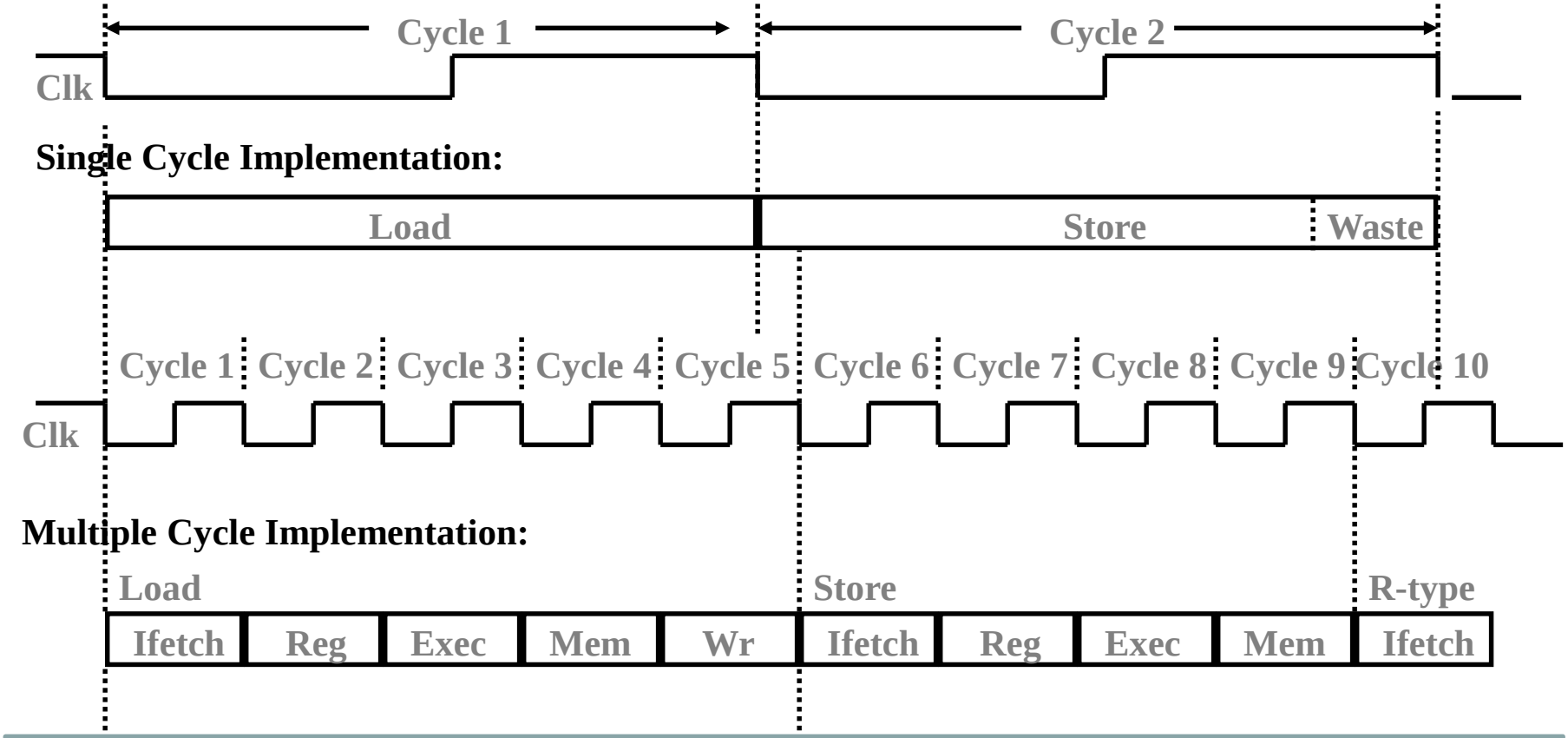
Exceções no RISC V



Exceções no RISC V



Mono, multi-ciclo



Projeto

- Projetar CPU que implementa repertório de uma CPU similar ao RISC V
- Dados módulos em Verilog para cada componente da Unidade de processamento (ALU, Banco de Registradores, PC, Memória, etc...)
- Projetar unidade de processamento pela interligação dos módulos
- Projetar unidade de controle

Instrução	Descrição
nop	No operation
lui regd	regd= (32b-immed(31), immed, 12b0)
ld reg, desl(reg_base)	reg. = mem (reg_base+desl)
sd reg, desl(reg_base)	Mem(reg_base+desl) = reg
lw reg, desl(reg_base)	reg. (31:0)= mem [reg_base+desl](31:0) , extensão do sinal
sw reg, desl(reg_base)	Mem[reg_base+desl](31:0) = reg(31:0)
lh reg, desl(reg_base)	reg. (15:0)= mem [reg_base+desl](15:0) , extensão do sinal
sh reg, desl(reg_base)	Mem[reg_base+desl](15:0) = reg(15:0)
lbu reg, desl(reg_base)	reg. (7:0)= mem [reg_base+desl](7:0) , completa com zeros
sb (reg, desl(reg_base)	Mem[reg_base+desl](7:0) = reg(7:0)
add regi, regj,regk	Regi. <- Regj. + Regk
sub regi, regj, regk	Regi. <- Regj. – Regk
and regi, regj,regk	Regi. <- Regj. and Regk
addi regi, regj, cte	Regi = Regj + cte
srli regd, regs, n	Desloca regs para direita n vezes sem preservar sinal, armazena valor deslocado em regd
srai regd, regs, n	Desloca regs para dir. n vezes preservando o sinal, armazena valor deslocado em regd.
slli regd, regs, n	Desloca regs para esquerda n vezes, armazena valor deslocado em regd.
slt regi, regj,regk	Regi = 1 se (regj) < (regk) caso contrário regi= 0
slti regi, regj,immed	Regi = 1 se (regj) < extensão sinal de immed, caso contrário regi= 0
beq regi, regj, desl	PC = PC + desl*2 se regi = regj
bne regi, regj, desl	PC = PC + desl *2 se regi <> regj
bge regi, regj, desl	PC = PC + desl*2 se regi >=regj
blt regi, regj, desl	PC = PC + desl*2 se regi < regj
jal ra,end	ra = pc, pc = pc+ end * 2, se Ra=0 não guarda PC
jalr ra, desl(reg-dst)	Ra=Pc Pc= reg-dst+desl, se Ra=0 não guarda PC
break	Para a execução do programa